ALGORITMA DAN PEMROGRAMAN FREE PASCAL

ROBIN, S.KOM., M.TI., M.TA
ALI AKBAR LUBIS, S.KOM., M.TI.
IRWAN BUDIMAN, S.T., M.T.

BORLAND TURBO

ALGORITMA DAN PEMROGRAMAN FREE PASCAL

Penulis

Robin,

Ali Akbar Lubis,

Irwan Budiman

Editor

Ali Akbar Lubis

Penerbit:
UNPRI PRESS

Redaksi

Jl. Sampul, Medan

ISBN:

Kata Pengantar

Saat buku ini ditulis, perkembangan teknologi perangkat lunak mengalami perkembangan yang sangat pesat, baik pada sektor industri perangkat lunak, game ataupun perangkat lunak yang berbasis aplikasi bisnis dan layanan masyarakat, baik perangkat lunak berbasis komputer desktop/laptop, web ataupun mobile. Perkembangan perangkat lunak tersebut tidak terlepas dari infrastruktur (engine) perangkat lunak dimana engine prangkat lunak tersebut dikembangkan melalui algoritma, cara berpikir logis dan cara penyelesaian secara matematis dan sistematis yang dituangkan lewat pemrograman komputer sehingga perangkat lunak canggih dapat dikembangkan dari engine perangkat lunak tersebut. Bahkan saat buku ini ditulis, sudah ada programming class yang diselenggarakan secara online untuk siswa-siswa sekolah dasar dan telah ada mata pelajaran programming sebagai mata pelajaran pilihan pada sekolah-sekolah terkemuka di kota-kota besar di tingkat SLTA.

Selain industri perangkat lunak dan game, bidang ilmu lainnya juga sangat bergantung kepada algoritma dan pemrograman. Berikut ini adalah bidang ilmu yang membutuhkan pemrograman di dalam pengembangannya.

1. Teknik Elektro (Electrical Engineering)
2. Teknik Komputer (Computer Engineering)
3. Sistem Komputer (Computer System)
4. Teknik Informatika (Computer Science/Informatics)
5. Sistem Informasi (Information System)
6. Rekayasa Perangkat Lunak (Software Engineering)
7. Multimedia dan Komunikasi Visual
8. Teknik Telekomunikasi (Telecommunication Engineering)

Buku ini ditulis dengan melihat kebutuhan akan keahlian dalam memrogram algoritma dalam dunia kerja terutama pada 8 bidang yang dituliskan di atas dengan harapan dapat memenuhi kompetensi sumber daya manusia sehingga di masa depan dengan menguasai algoritma dan pemrograman dapat mempermudah proses penerimaan pegawai pada dunia

kerja dan memiliki kesempatan yang besar untuk diterima pada perusahaan pengembangan aplikasi berbasis komputer.

Dengan dipublikasikannya buku ini tim penulis ingin memberikan ucapan terima kasih secara khusus kepada:

1. **Bapak Okky Putra Barus, S.Kom., M.Ti.** dan **Ibu Ferawaty, S.Kom., M.Kom.** selaku kepala program studi Sistem Informasi dan program studi Informatika Universitas Pelita Harapan Medan atas bantuan dan dukungannya dalam mempublikasikan buku ini.
2. Ibu **Anita Christine Sembiring, S.T., M.T.** atas dukungannya dalam menggunakan bahasa pemrograman Delphi dan Pascal pada mata kuliah pemrograman di dalam program studi Teknik Industri pada Universitas Prima Indonesia sehingga penulis memiliki kesempatan untuk membantu Beliau dengan menulis dan mempublikasikan buku ini.

Medan, 28 Juli 2023

Tim Penulis

Daftar Isi

Daftar Gambar

Daftar Tabel

Pengantar - ASCII, Simbol, Operator, Pengenal, Variabel, Input dan Output dalam Bahasa Pemrograman Pascal

Manfaat Pembelajaran:

1. Mengetahui cara kerja sistem komputer, dimulai dari karakter ASCII, huruf latin dan simbol-simbol yang terdapat di dalam teknologi sistem komputer agar memudahkan dalam memrogram program komputer.
2. Mengetahui karakteristik dari bahasa pemrograman (khususnya bahasa pemrograman Pascal) dimulai dari pengenal, tipe data, variable, input dan output

Berikut ini adalah pernyataan yang membutuhkan penalaran logika.

Contoh 1:

Pernyataan 1: Semua tukang kayu adalah pria

Pernyataan 2: Ria bukan seorang pria

Kesimpulan: Ria bukan tukang kayu

Contoh 2:

Pernyataan 1: Semua pembantu rumah tangga adalah wanita

Pernyataan 2: Adi adalah seorang pria

Pernyattaan 3: Ida adalah seorang wanita

Kesimpulan yang pasti: Adi bukan pembantu rumah

Kesimpulan yang mungkin: Ida mungkin seorang pembantu rumah tangga

Contoh 3:

Pernyataan 1: Tidak semua karyawan rajin bekerja

Pernyataan 2: Budi adalah seorang karyawan

Kesimpulan: Budi rajin bekerja atau Budi malas bekerja ?

Contoh 4:

Pernyataan 1: Sebuah mata pelajaran sekolah dinyatakan lulus apabila nilai ulangan lebih besar atau sama dengan 60

Pernyataan 2: Nilai ulangan matematika Budi adalah 50

Kesimpulan: Budi tidak lulus pada mata pelajaran matematika

Berikut ini adalah contoh penggunaan logika dalam dunia digital.

1. Benar atau Salah
2. Hidup atau Padam (On/Off)
3. 0 atau 1
4. Dapat dioperasikan dengan operator logika dasar AND, OR dan NOT

Berikut ini adalah definisi dan sifat-sifat algoritma.

1. Dirancang secara sistematis dari cara berpikir secara logis.
2. Merupakan langkah-langkah sistematis dan berurutan yang harus dilakukan untuk menyelesaikan suatu masalah komputasi.
3. Membutuhkan bantuan variabel dengan struktur data dengan tipe data tertentu untuk menyelesaikan masalah komputasi.
4. Menyelesaikan permasalahan komputasi dengan menguraikan langkah-langkah penyelesaian secara berurutan dan sistematis (berorientasi problem solving).

Contoh Algoritma dalam Kehidupan Sehari-hari: **bangun pagi**.

1. Membuka mata
2. Bangkit dari tempat tidur

3. Berdiri tegak
4. Jalan keluar dari kamar tidur

Contoh Algoritma dalam Kehidupan Sehari-hari: **berangkat kerja dari rumah di pagi hari**.

1. Bangun dari tempat tidur.
2. Jika sehat maka lanjut ke langkah ketiga.
   Jika kurang sehat maka tidur kembali dan hubungi HRD bahwa sedang sakit dan tidak masuk kerja dan langkah-langkah di bawah ini tidak perlu dilakukan.
3. Menggosok Gigi dengan …
4. Mandi di …
5. Mengkonsumsi sarapan berupa …
6. Berangkat kerja dengan naik …
7. Naik ke ruangan kantor pada lantai … dengan menggunakan …

Contoh Algoritma dalam Kehidupan Sehari-hari: **menggosok gigi**.

1. Mengambil sikat gigi.
   a. Jika sikat gigi rusak maka ambil yang baru yang tersedia.
   b. Jika sikat gigi yang baru tidak tersedia maka gunakan kembali sikat gigi yang rusak atau tidak sikat gigi sama sekali.
2. Mengoleskan pasta gigi.
   a. Jika pasta gigi habis maka ambil pasta gigi yang baru.
   b. Jika tidak ada pasta gigi yang baru maka keluar dari rumah menunju ke mini market sebelah untuk beli pasta gigi.
   c. Jika tidak ada mini market maka tidak jadi gosok gigi.

Berikut ini adalah definisi pemrograman komputer.

1. Merupakan implementasi dari logika dan algoritma.
2. Sekumpulan instruksi program yang yang membentuk sebuah proses.
3. Sekumpulan instruksi program digunakan untuk menyelesaikan masalah komputasi berbasis problem solving.
4. Dengan melalui proses kompilasi maka program yang dirancang akan dikonversi menjadi program berbahasa mesin yang dapat dimengerti oleh sistem operasi (Operating System/OS) computer.

Kata **algoritma** berasal dari bahasa Persia yang pertama kali diperkenalkan oleh seorang matematikawan Persia bernama Muḥammad bin Musa al-Khwarizmi yang selanjutnya sering disebut sebagai Al-Khwarizmi. Al-Khwarizmi dikenal sebagai bapak Al-jabar (Al-Zebra) karena berhasil menyelesaikan persamaan matematika dengan cara menjabarkannya ataupun memindahkan dengan cara dan aturan matematika tertentu. Selain menemukan Al-Jabar, Al-Khwarizmi juga menemukan cara menyelesaikan permasalahan komputasi dengan teknik Al-Jabar yang sekarang disebut sebagai algoritma.

Sebelum dapat merancang, mengembangkan algoritma dengan baik dan sebelum dapat memrogram algoritma ke dalam salah satu bahasa pemrograman dengan baik, pemahaman dan kemampuan dalam menyelesaikan permasalahan Al-Jabar (Al-Zebra) sangat dibutuhkan.

Berikut ini adalah beberapa contoh latihan pemahaman Al-Jabar yang sangat membantu dalam mempelajari algoritma dan pemrograman.
Kategori 1: Bagaimana cara mendapatkan nilai variabel x dari persamaan berikut ini.

1. 5x=90+10
2. 100=x/10-5

3. 90=30+5x+10

Kategori 2: Ada berapa cara (uraian) yang dapat dilakukan untuk operasi aritmatika berikut agar lebih mudah dihitung tanpa bantuan alat hitung.

4. 45 + 38
   - 40 + 30 + (5+8) = 70 + 13 = 83
   - 45 + 38 + 0 = 45 + 38 - (35-35) = 45 + 38 +35 - 35 = 45 + 35 + 38 -35 = (45 + 35) + (38 -35) = 80 + 3 =83
5. 76 * 12
   - 70 * 12 + 6 * 12 = 840 + 72 = 912
   - 76 * 10 + 76 *2 = 760 + 152 = 912

Bahasa Pemrograman Komputer (Computer Programming Language) merupakan sebuah bahasa yang bekerja secara elektronik yang memiliki aturan-aturan yang harus dipatuhi oleh pemrogram, yang dirancang sedemikian rupa agar pemrogram (programmer) dapat merancang program dengan lebih mudah. Bahasa perograman komputer dapat dikategorikan sebagai berikut berdasarkan arah dan tujuan dari penggunaan bahasa pemrograman tersebut.

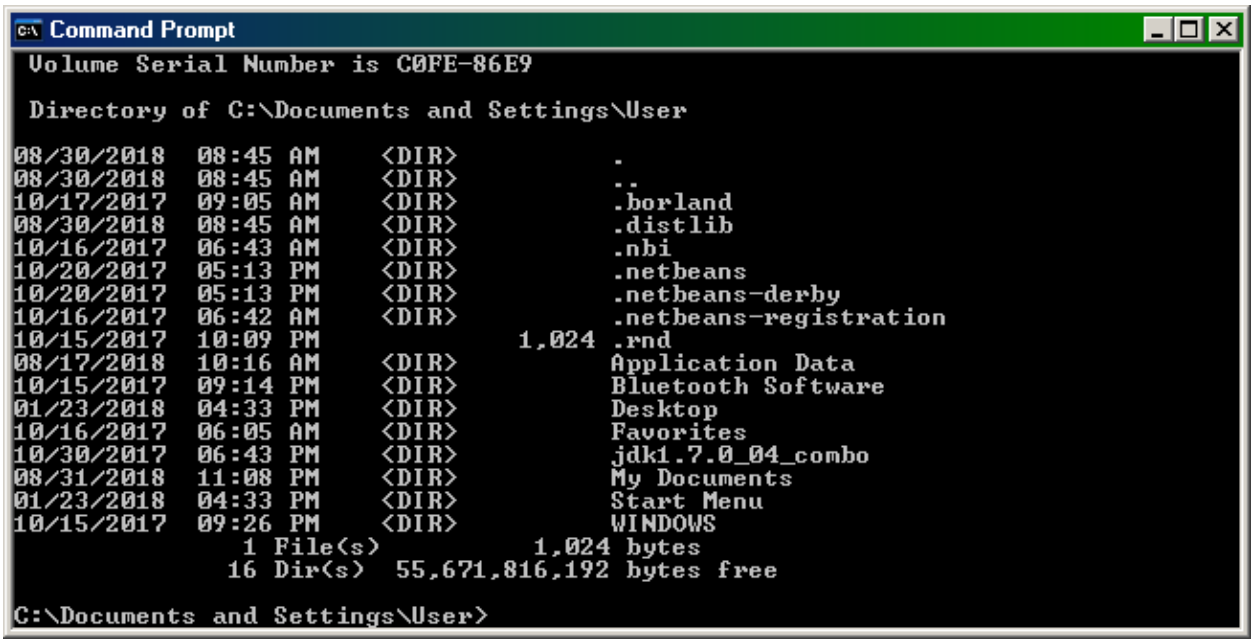
1. Low Level Language (bahasa pemrograman yang lebih dekat dengan bahasa mesin)
2. Mid Level Language
3. High Level Language (bahasa pemrograman yang lebih dekat dengan bahasa manusia)

Program vs Software:

Program komputer adalah sebuah realisasi dari konsep penyelesaian masalah komputasi yang berbentuk algoritma ataupun prototype solution yang merupakan bagian kecil dari sebuah perangkat lunak sedangkan software (perangkat lunak) adalah sebuah produk aplikasi komputer yang di dalamnya terdapat banyak solusi yang sudah diimplementasikan dengan sekumpulan program komputer dalam cakupan bidang yang sama.

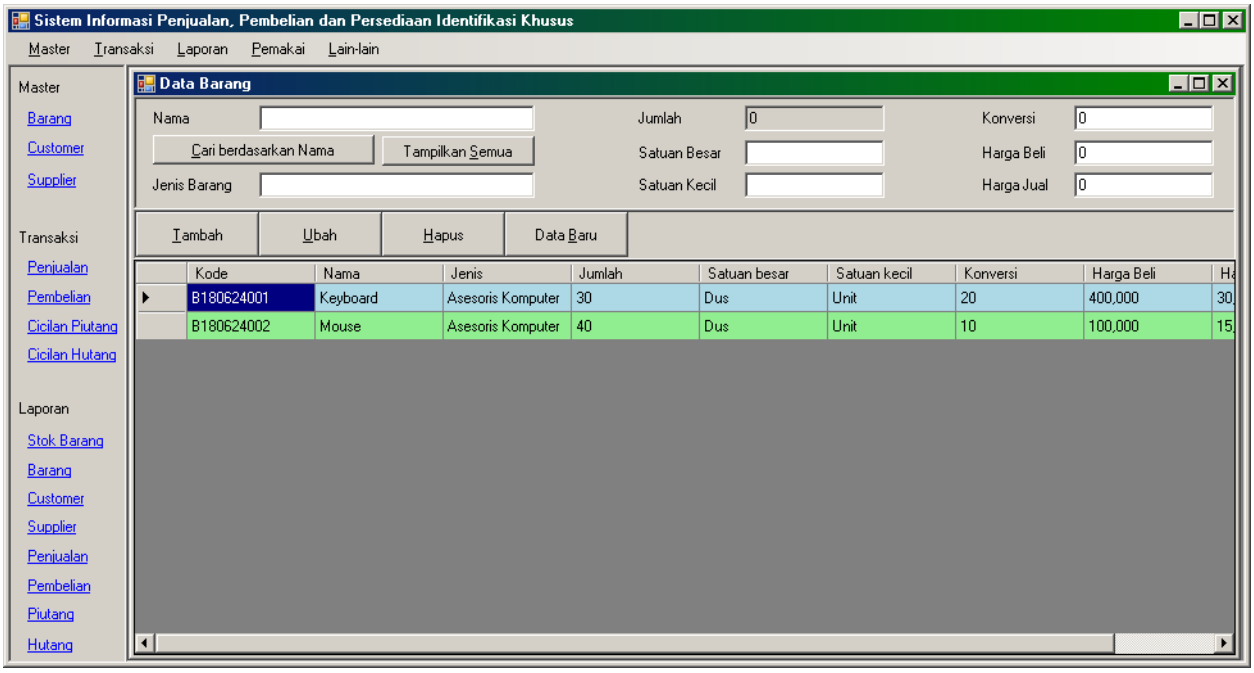
Berikut ini adalah platform pemrograman dan perangkat lunak yang populer hingga saat buku ini ditulis.

1. Console: Desktop/Laptop (Contoh: DOS dan Unix)



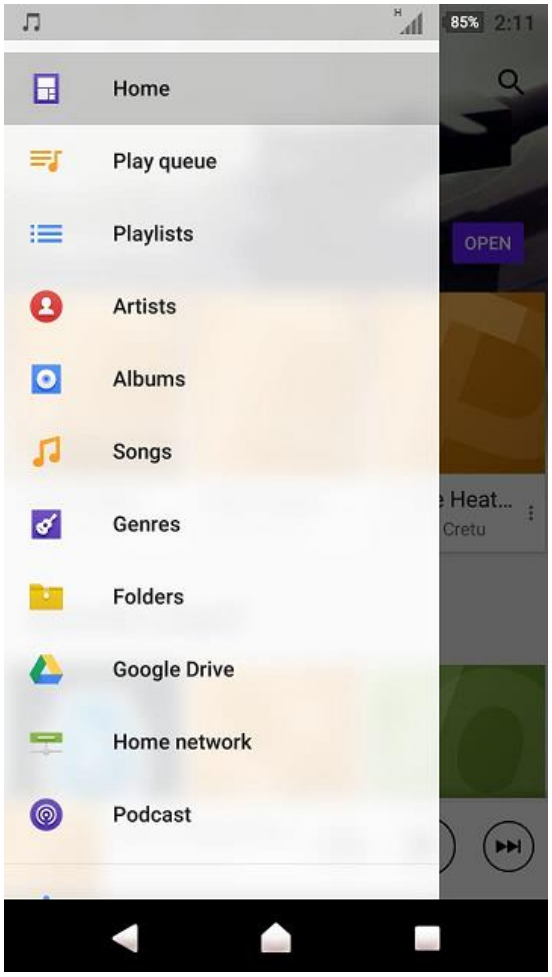
Gambar 1 Platform Console

2. Graphical User Interface (GUI).
    a. Desktop/Laptop (Contoh: Windows, Mac OS, Linux, dll)



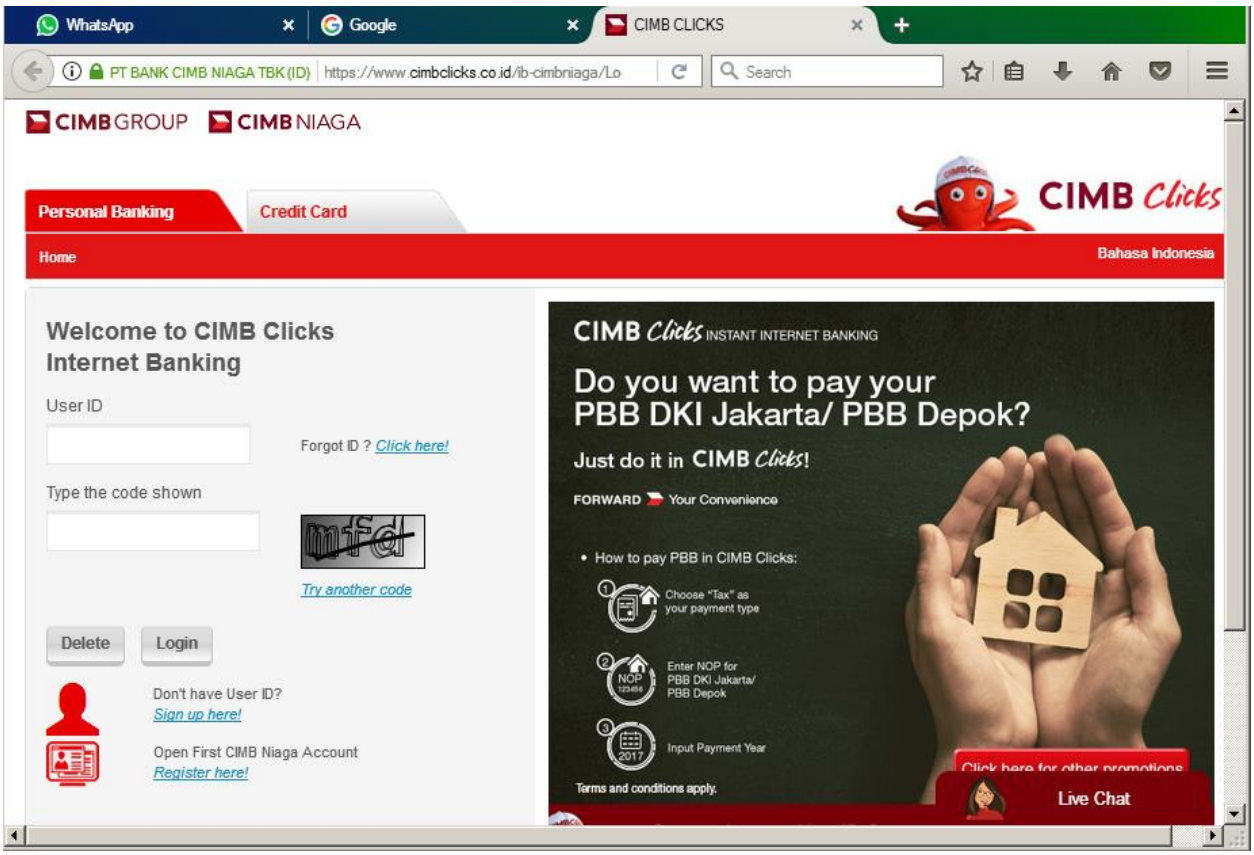
Gambar 2 Platform Desktop GUI

b. Portable Devices (Contoh: Android, iOS, PSP, dll)



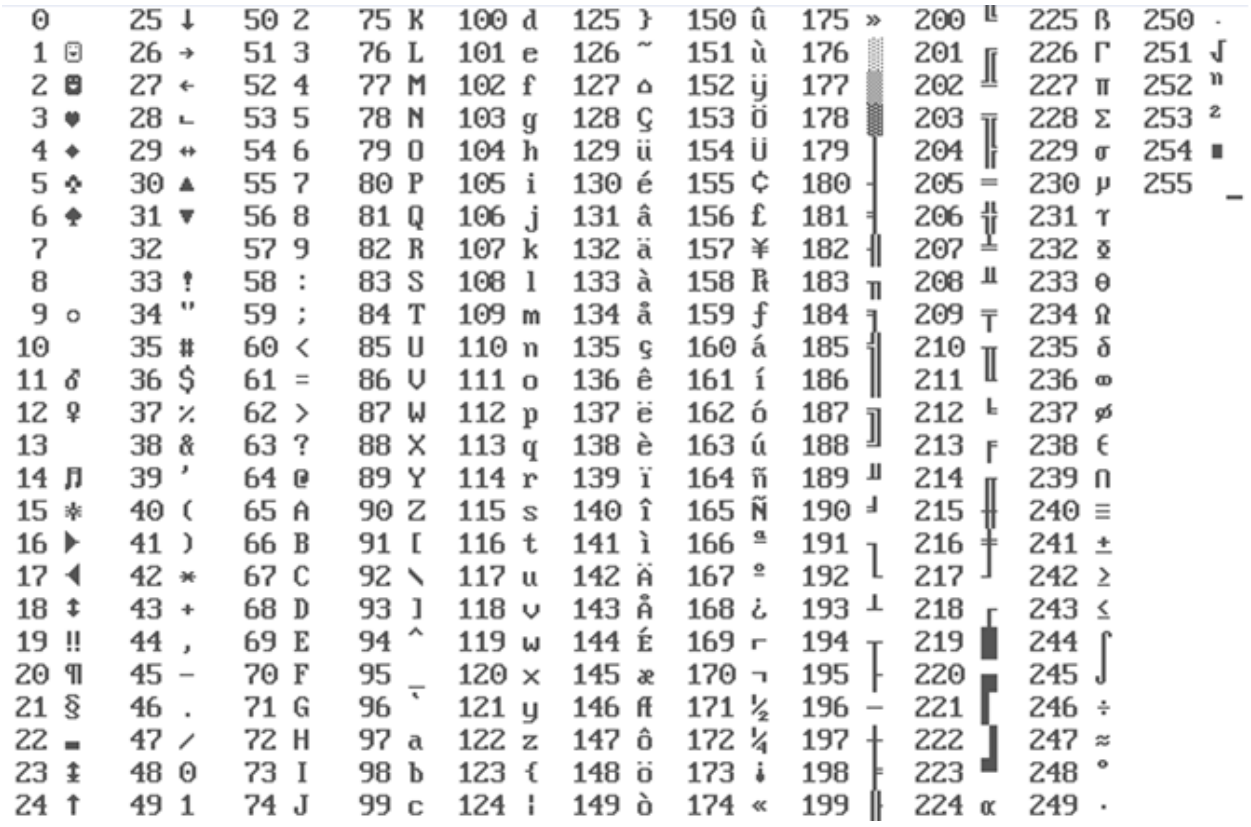
Gambar 3 Platform Mobile

3. Web



Gambar 4 Platform Web

ASCII adalah kepanjangan dari American Standard Code for International Interchange. ASCII pertama kali digunakan oleh perusahaan komputer raksasa yang bernama IBM (International Bussiness Machine) dimana perusahaan tersebut saat buku ini ditulis telah diakusisi oleh perusahaan industri komputer yang bernama Lenovo. Alasan dibuatnya kode ASCII ini sangat sederhana, yaitu karena komputer merupakan peralatan elektronik yang tujuan utamanya adalah untuk berhitung, maka awalnya komputer hanya bisa mengolah data-data numerik dan tidak memiliki abjad latin sehingga kode ASCII dibuat agar manusia menjadi lebih mudah berinteraksi dengan komputer dengan menggunakan abjad latin dan tanda baca daripda harus menggunakan bahasa mesin dan kode-kode heksa desimal dalam mengoperasikan komputer. Setiap byte dari kode ASCII yang memiliki nilai byte dari 0 s/d 255 dipetakan menjadi huruf abjad latin 'A' s/d 'Z' dan 'a' s/d 'z', angka '0' s/d '9', tanda baca dan beberapa simbol lainnya yang mungkin dibutuhkan di dalam antar muka komputer dengan manusia (computer-human interface).



Gambar 5 Tabel ASCII versi IBM PC

Simbol (Symbol)

Sebuah simbol adalah sebuah karakter ASCII yang dikenal oleh bahasa pemrograman untuk mewakili sebuah kode ataupun tanda yang memiliki arti tertentu bagi sebuah bahasa pemrograman. Sebuah simbol dapat berupa sebuah tanda memulai dan mengakhiri komentar, sebuah tanda kurung, operator aritmatika, operator perbandingan ataupun sebuah operator logika.

Berikut ini adalah contoh simbol dan artinya dalam beberapa bahasa pemrograman.

1. # adalah sebuah simbol komentar di dalam bahasa pemrograman Python.
2. ‘ adalah sebuah simbol komentar di dalam bahasa pemrograman BASIC dan keturunannya.
3. REM adalah sebuah pengenal yang berupa komentar di dalam bahasa pemrograman BASIC dan keturunannya.
4. {, }, (*, *) adalah beberapa contoh simbol pembuka dan penutup komentar di dalam bahasa pemrograman Pascal dan keturunannya.

5. // adalah sebuah simbol komentar di dalam bahasa pemrograman C dan keturunannya.
6. /*, */ adalah beberapa contoh simbol pembuka dan penutup komentar di dalam bahasa pemrograman C dan keturunannya.
7. +, -, *, /, %, = adalah beberapa simbol yang merupakan operator aritmatika yang paling umum terdapat di dalam hampir semua bahasa pemrograman.
8. +=, -=, /=, *=, %= adalah beberapa symbol yang merupakan operator increment dan decrement yang terdapat di beberapa bahasa pemrograman.

Pengenal (Identifier)

Sebuah pengenal harus memiliki sebuah nama yang unik (tidak boleh ada 2 atau lebih pengenal dengan nama yang sama) sebagai sebuah identitas yang dikenal di dalam sebuah program yang menyebabkan pengenal tersebut dapat dibedakan dengan pengenal lainnya. Sebuah pengenal dapat berupa sebuah perintah program, tipe data, konstanta, variabel ataupun sebuah fungsi. Pengenal merupakan sebuah kata yang dikenal oleh sebuah bahasa, baik bahasa manusia ataupun bahasa komputer. Sebuah pengenal dapat berupa kata kunci, perintah program, konstanta, variabel atau sub program.

Sebuah pengenal dalam bahasa pemrograman memiliki persyaratan umum sebagai berikut.

1. Hanya boleh berupa 1 kata dan hanya boleh terdiri dari huruf, angka dan garis bawah
2. Hanya boleh diawali dengan huruf dan garis bawah (underscore) dan tidak boleh diawali dengan angka
3. Tidak boleh menggunakan karakter khusus seperti spasi dan tanda ~, `, !, @#, $, %, ^, &, *, (,), +, -, /, “, ‘, :, ;, ?, ., <, > dll
4. Hampir semua bahasa pemrograman komputer membedakan pengenal dengan penulisan yang sama persis tetapi berbeda pada hanya huruf besar dan kecil. Misalnya variabel **Nama** akan dianggap berbeda dengan variable **nama** dan **NAMA** kecuali bahasa pemrograman sekelas BASIC (dan keturunannya) dan Pascal (dan keturunannya, salah satunya yaitu Delphi).

Konstanta (Constant) vs Variabel (Variable)

Konstanta adalah sebuah pengenal dimana isi dari pengenal tersebut adalah bersifat tetap dan tidak dapat diubah sedangkan variabel adalah sebuah pengenal dimana isi dari pengenal tersebut adalah bersifat tidak tetap dan dapat diubah. Konstanta dan variable dapat bertipe bilangan bulat (short integer, integer, long integer), pecahan (single precision dan double precision ataupun extended atau long double precision), character atau huruf, string atau kalimat. Sebuah konstanta dapat diwakili oleh sebuah kata, misalnya konstanta **3.14152** diwakili oleh konstanta pengenal **PI**, konstanta string **"Pesan Error"** diwakili oleh konstanta pengenal **ErrMsg**. Sedangkan variabel sudah pasti diwakili oleh sebuah kata sebagai pengenal pada sebuah bahasa pemrograman.

Tipe Data Dasar

Berikut ini adalah tipe data dasar yang digunakan pada bidang matematika.

1. Bilangan Bulat
    a. Bilangan Cacah
    b. Bilangan Asli
    c. Bilangan Positif
    d. Bilangan Negatif
2. Bilangan Pecahan
    a. Pecahan Biasa
    b. Pecahan Campuran
    c. Pecahan Desimal

Sedangkan yang berikut ini adalah tipe data dasar yang digunakan di dalam pemrograman komputer.

1. Bool (Logika)
    a. 0 atau 1
    b. Salah atau Benar (True/False)
    c. Padam atau Hidup (On/Off)
2. Numerik
    a. Bilangan Bulat (Byte, Short Integer, Integer, Long Integer, Integer 16 bit, Integer 32 bit, Integer 64 bit)
    b. Bilangan Pecahan (floating or real data type, Single Precision, Double Precision, Extended)
3. Non-Numerik
    a. Character (char) sebagai tipe data huruf atau kode ASCII
    b. String (sebagai tipe data kalimat dimana pada bahasa pemrograman tertentu, tipe data string tidak dianggap sebagai tipe data dasar)

Berikut ini adalah tipe data lanjutan dalam pemrograman komputer.

1. Array/Deret
2. Record/Structure
3. Date & Time
4. Variant
5. Class/Object
6. dll

Berikut ini adalah metode penyelesaian masalah pemrograman komputer.

1. Rumuskan permasalahan menjadi input dan output.
2. Buat sketsa tampilan input dan output program (program interface prototype) agar memiliki gambaran mengenai bagaimana program komputer akan menerima data-data

yang akan dimasukkan ke dalam program (input) dan bagaimana program komputer akan memberikan (mencetak) hasil operasi program (output).

3. Karena alur program komputer secara umum adalah terdiri dari input – proses – output maka susun kerangka (prototype) langkah-langkah penyelesaian program komputer secara konsep dalam bentuk algoritma.
4. Tulis dan implementasikan input, langkah-langkah penyelesaian dan output ke dalam salah satu bahasa pemrograman komputer.

Output sebuah program komputer adalah sebuah target, sebuah gambaran, sebuah penglihatan, sebuah hasil yang diharapkan untuk diselesaikan oleh program komputer dengan benar pada akhir dari sebuah program. **Input** sebuah program komputer merupakan hal/data apa saja yang dibutuhkan oleh komputer untuk menyelesaikan permasalahan komputasi dan menghasilkan output. Input komputer dapat berupa dapat berupa data atau informasi yang dibutuhkan oleh program untuk menyelesaikan permasalahan komputasi dan menghasilkan output program, dapat juga merupakan hasil komputasi dari program lain atau peralatan komputer elektronik untuk diolah oleh program komputer.

Ada 4 model penulisan pemecahan permaslahan komputer.

1. Algoritma

   Algoritma dapat ditulis dengan menggunakan bahasa manusia apapun, baik bahasa nasional ataupun bahasa daerah.
2. Pseudo code

   Pseudo code adalah sebuah standar penulisan modal/prototype langkah-langkah penyelesaian permasalahan komputasi secara konsep yang harus ditulis dalam bahasa Internasional (bahasa Inggris).
3. Flowchart

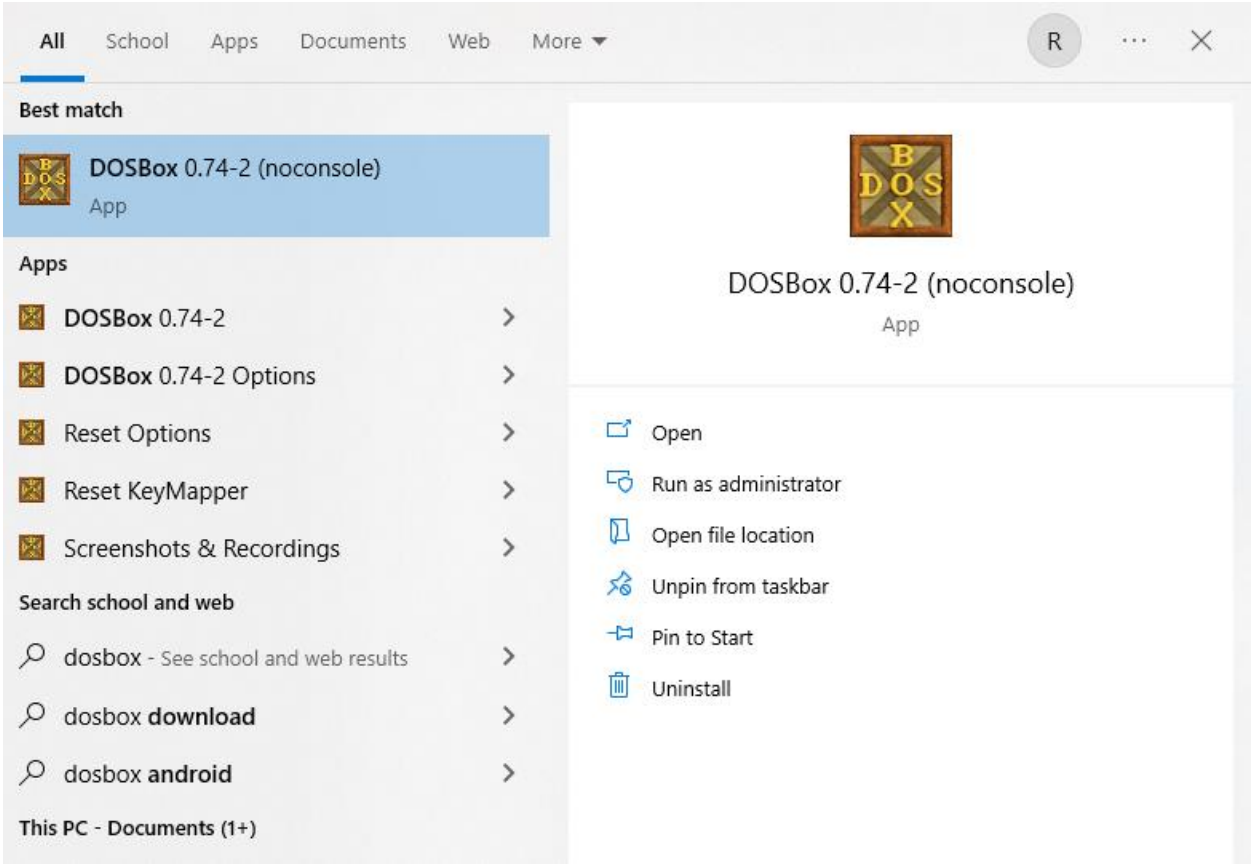
   Flowchart adalah sebuah diagram standar yang dapat digunakan untuk menggantikan algoritma ataupun pseudo code.
4. Bahasa program komputer

Penulisan pemecahan permaslahan komputer dapat juga langsung diimplementasikan ke dalam salah satu bahasa pemrograman untuk langsung dipraktekkan dan diuji coba.

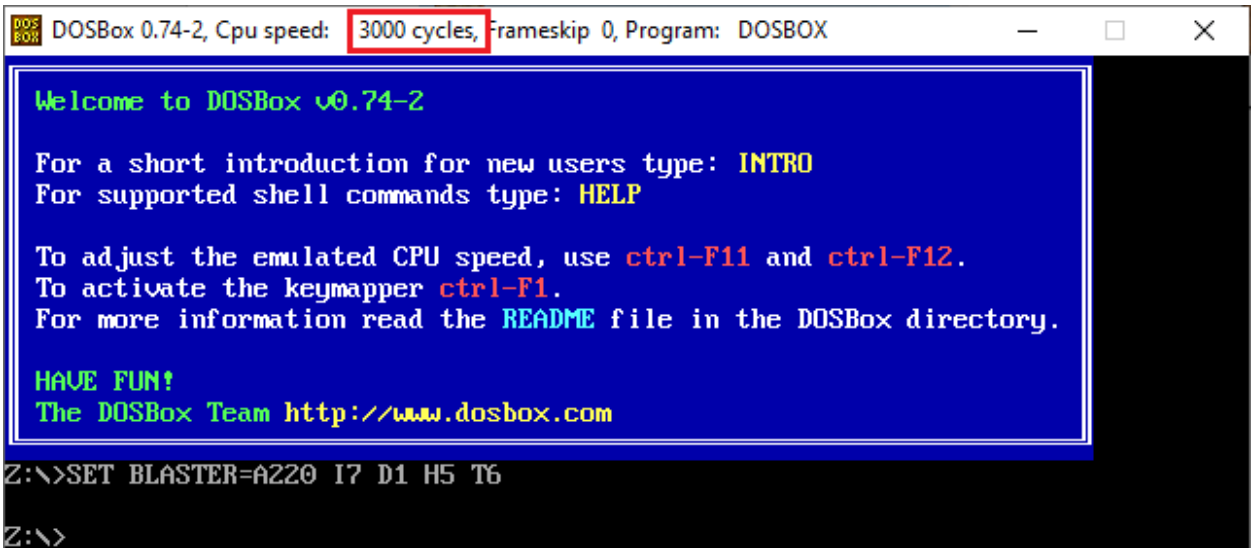
Ada 4 tool bahasa pemrograman Pascal yang dapat digunakan yaitu Turbo Pascal, Free Pascal, Borland Delphi dan RAD Studio. Turbo Pascal hanya dapat dijalankan pada sistem operasi DOS atau melalui emulator DOSBox sementara Free Pascal, Borland Delphi dan RAD Studio hanya tersedia versi Windows. RAD Studio memiliki kelebihan dapat membuat program untuk sistem operasi Linux meskipun RAD Studio sendiri hanya dapat berjalan pada sistem operasi Windows.

Cara memasang dan menjalankan Turbo Pascal:

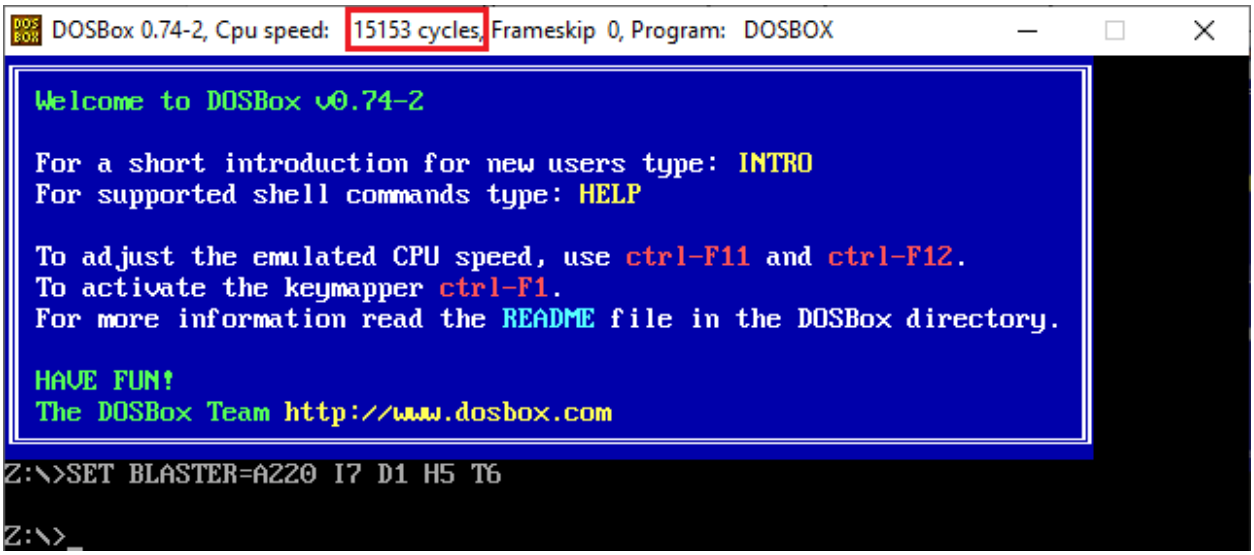
1. Pastikan Turbo Pascal telah ada pada komputer, misalnya pada buku ini Turbo Pascal terpasang pada folder D:\COMPILER\TP71 dimana program dimulai dari menjalankan Turbo.exe di dalam sub folder bin dari folder ini.
2. Download dan Install DOSBox dari https://www.dosbox.com/download.php?main=1 karena Turbo Pascal adalah Bahasa pemrograman yang hanya bisa dijalankan pada sistem operasi DOS dan tidak bias dijalankan pada sistem operasi MacOS, Windows ataupun Linux.
3. Jalankan DOSBox Emulator



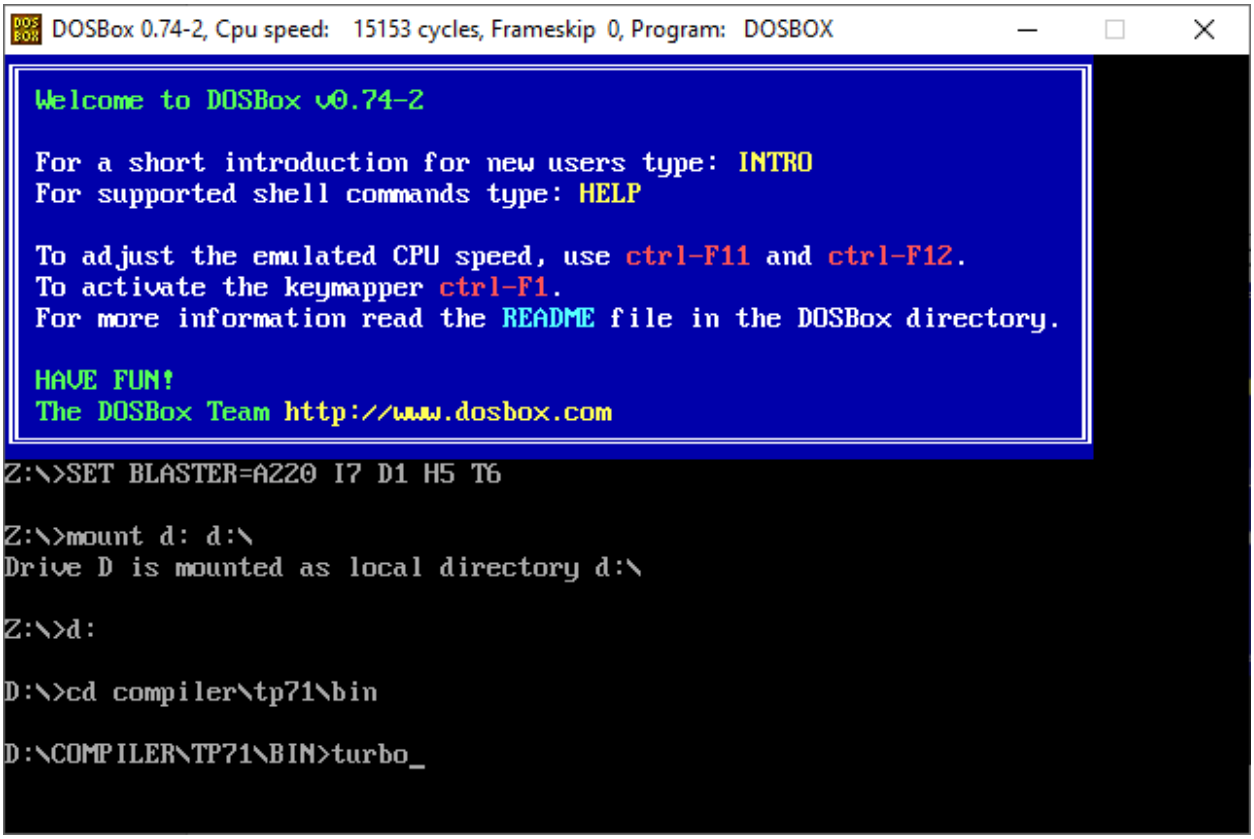
4. Agar emulator bekerja lebih cepat maka naikkan cycle dari 3000



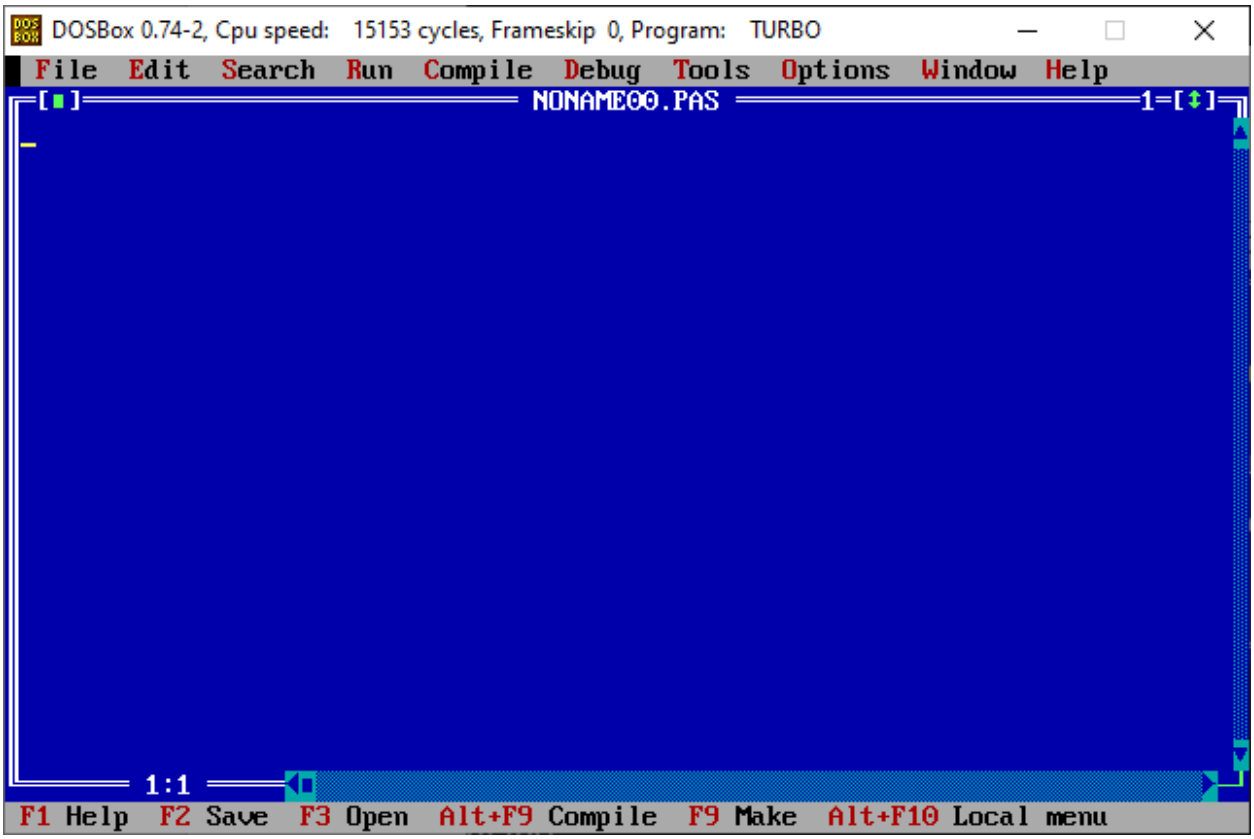
menjadi minimal 15000



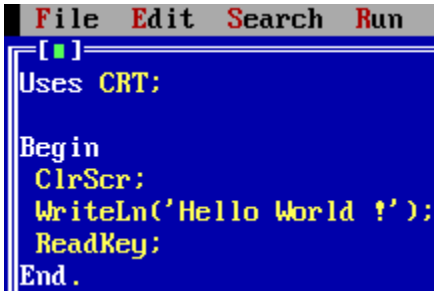
5. Jika Turbo Pascal terpasang pada folder seperti yang dijelaskan pada point 1 maka silahkan mount drive D seperti gambar berikut ini dan silahkan pindah folder ke D:\COMPILER\TP71\BIN dan jalankan Turbo.exe



6. Tampilan Turbo Pascal pada DOSBox

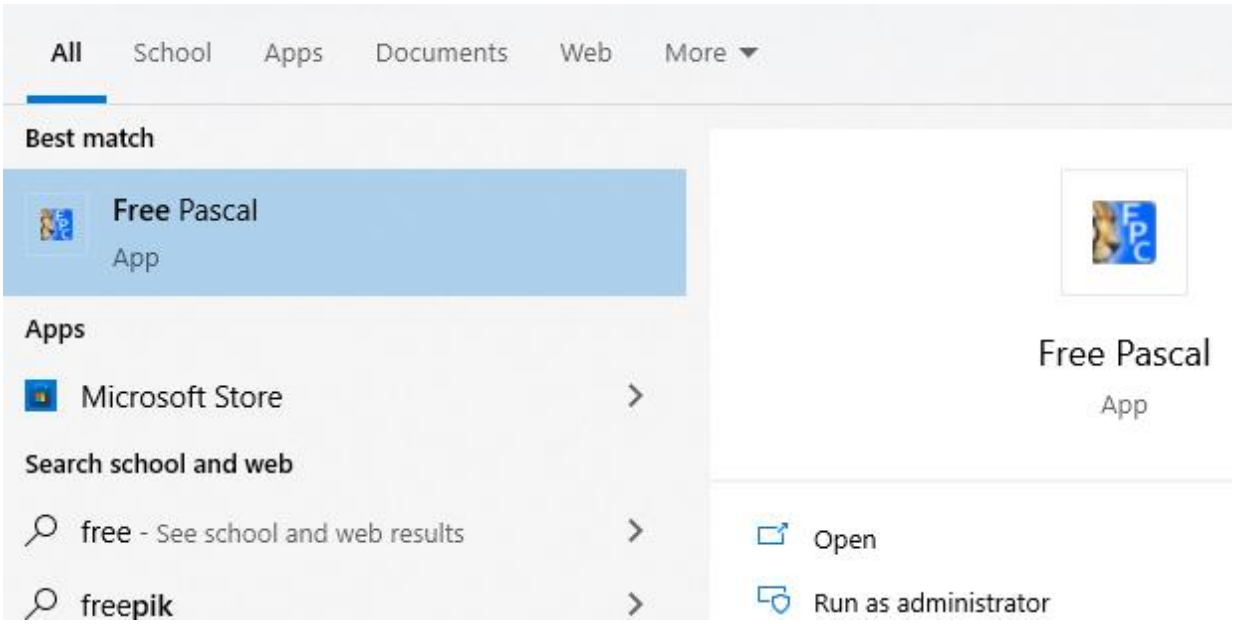


7. Ketikkan kode program ini dan jalankan program dengan membuka menu Run -> Run



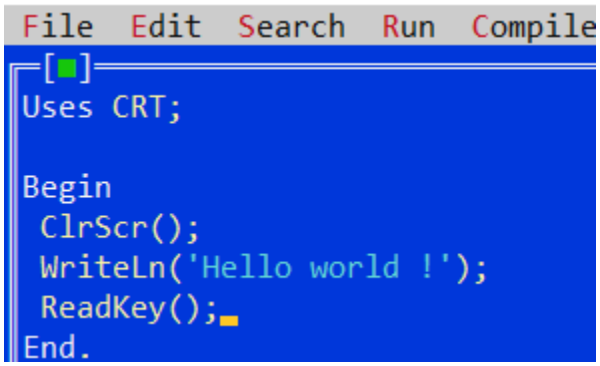
Cara memasang Free Pascal:

1. Download Free Pascal pada link ini https://www.freepascal.org/download.html kemudian lakukan proses install dari file yang telah berhasil di-download.
2. Jalankan Free Pascal dari start menu atau search



3. Free Pascal Berhasil dijalankan

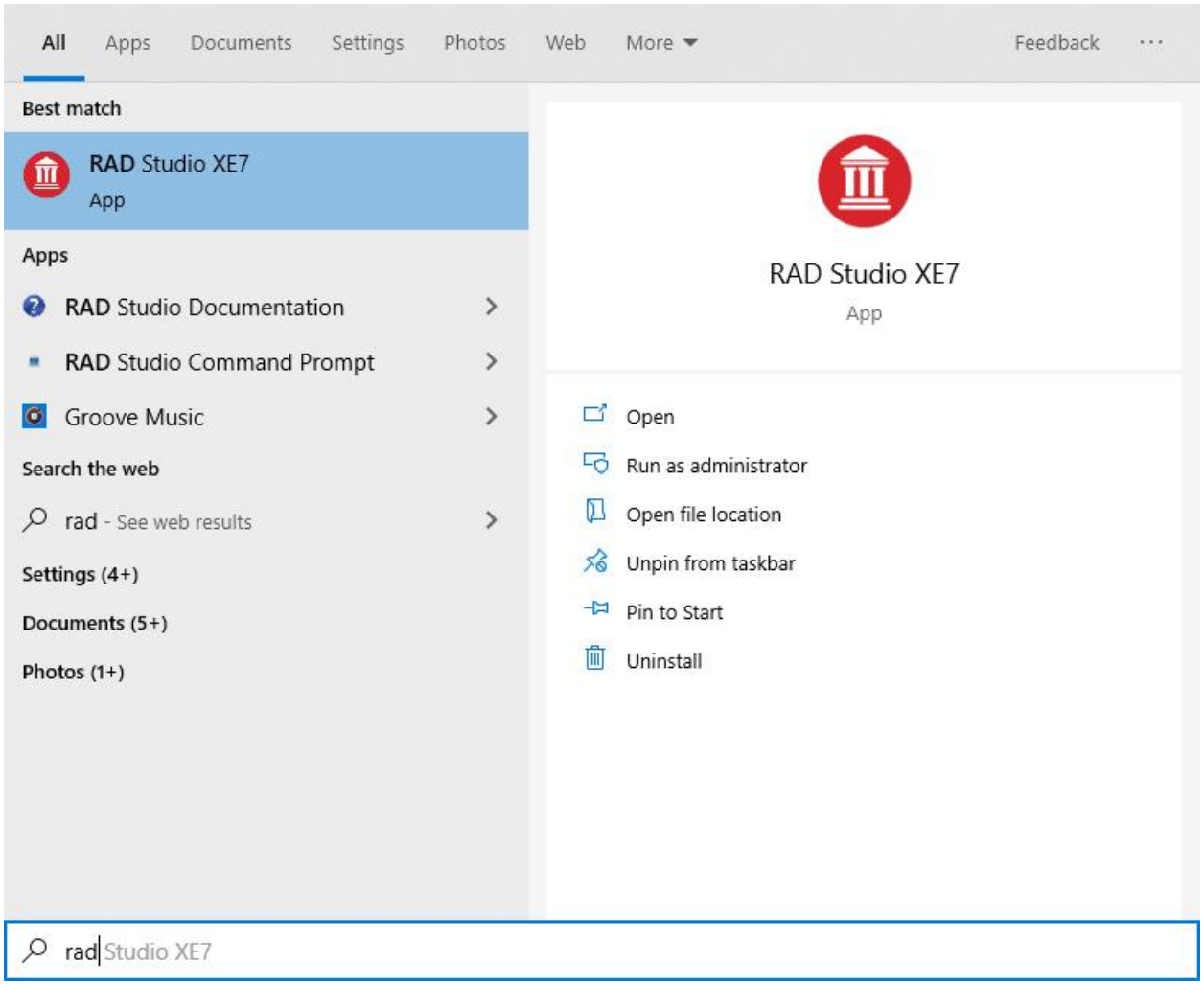
4. Buka menu File -> New dan ketikkan program berikut ini



5. Jalankan program dengan menekan tombol Ctrl + F9 atau menu Run -> Run

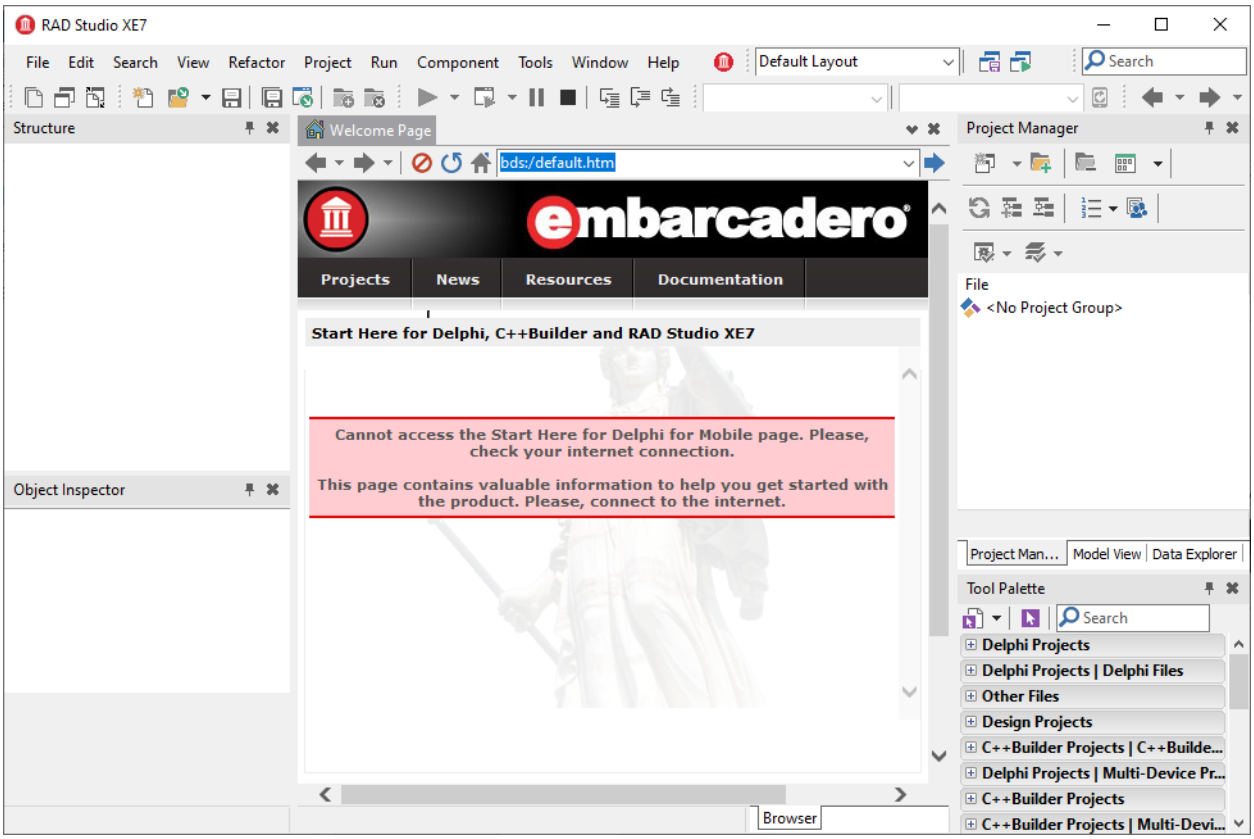
Cara menjalankan Delphi dari development tool RAD Studio dan memulai sebuah Console Application.

1. Cari RAD Studio dari Start Menu Windows. Bisa dengan mengetikkan huruf RAD. Pada screenshot di bawah ini penulis menggunakan RAD Studio versi XE7.

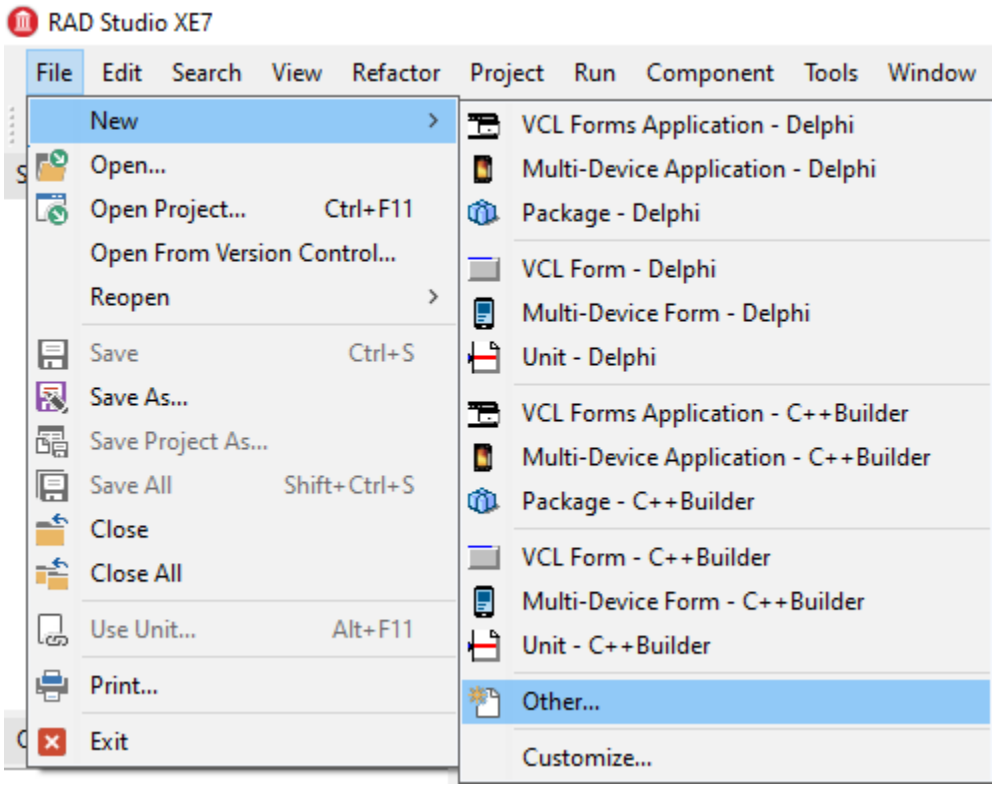


2. Tunggu proses loading saat logo RAD Studio baru muncul.

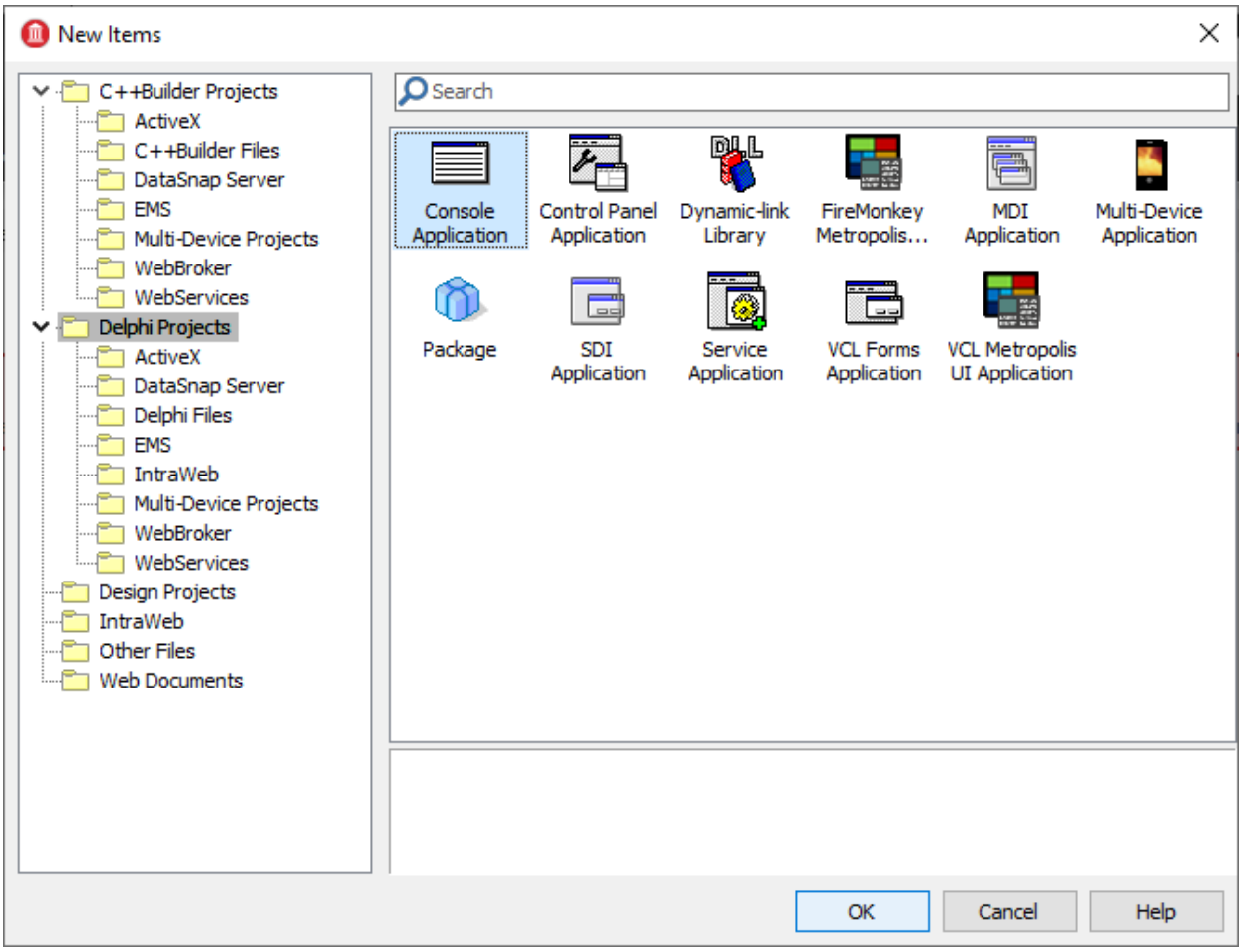
3. Jika proses loading sudah selesai maka tampilan RAD Studio akan terlihat seperti ini



4. Buka menu File -> New -> Other



5. Pilih Delphi Projects pada bagian kiri dan Console Application pada bagian tengah lalu tekan OK



6. Sebuah project program dengan beberapa code Delphi sudah disiapkan oleh RAD Studio

```
program Project1;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  System.SysUtils;

begin
  try
    { TODO -oUser -cConsole Main : Insert code here }
  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```

7. Perintah **try – except – end** yang secara otomatis di-generate oleh RAD Studio adalah sebuah perintah yang digunakan untuk mencegah terjadinya runtime error dengan cara meletakkan kode program yang diduga dapat menyebabkan runtime error di bagian **try**, lalu jika ada perintah program tertentu yang mau kita jalankan ketika terjadi error maka perintah program tersebut harus diletakkan pada bagian **on E: Exception do** yang terdapat pada bagian **except**. Namun pembahasan program pada buku ini kita sangat jarang menggunakan perintah **try – except – end** karena buku ini lebih berfokus kepada bagaimana mengembangkan algoritma dalam menyelesaikan masalah komputasi dan mengimplementasikannya ke dalam bahasa pemrograman Delphi sehingga perintah ini tidak akan dibahas secara spesifik pada buku ini. Untuk itu, maka bagian **try – except – end** ini kita hapus selama proses pembuatan program di buku ini.

```
program Project1;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  System.SysUtils;

begin
  try
    { TODO -oUser -cConsole Main : Insert code here }
  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```

8. Sehingga program kosong Delphi akan terlihat seperti gambar berikut ini dengan ditambahkan beberapa keterangan mengenai bagian deklarasi dan kode program

```
program Project1;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  System.SysUtils;

//deklarasikan konstanta, tipe data dan variabel global di sini

//deklarasikan function di sini

//deklarasikan variabel lokal yang digunakan oleh program utama
begin  //memulai program utama
 //semua perintah program utama di buat di sini
end.   //end. artinya akhir dari program utama
```

Berikut ini adalah contoh penyelesaian kasus menentukan status antara 2 buah bilangan apakah lebih besar atau tidak.

Dalam Algoritma

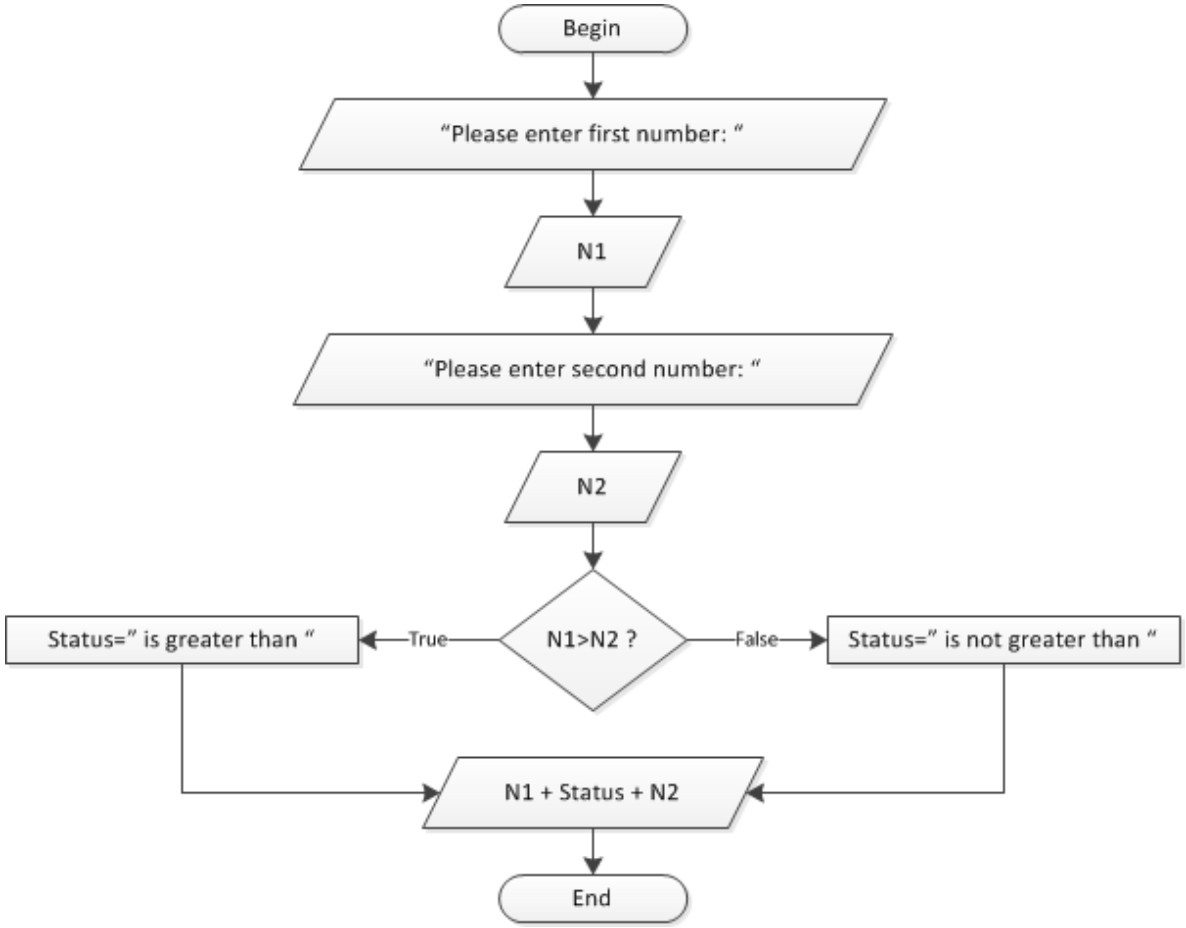
```
Cetak "Please enter first number: "
Masukkan N1
```

```
Cetak "Please enter second number: "
Masukkan N2
Apakah N1>N2 ?
Benar
 Status=" is greater than "
Salah
 Status=" is not greater than "
Cetak N1 + Status + N2
```

Dalam Algoritma

```
Display "Please enter first number: "
N1=Input
Display "Please enter second number: "
N2=Input
If N1>N2 Then
 Status=" is greater than "
Else
 Status=" is not greater than "
End If
Display N1 + Status + N2
```

Dalam Flowchart



Dalam Bahasa Pemrograman Turbo Pascal

```
uses CRT;

var N1, N2:Integer;
    Status:String;

begin
 Write('Please enter first number: ');
 ReadLn(N1);
 Write('Please enter second number: ');
 ReadLn(N2);

 if N1>N2 then
  Status:=' is greater than '
 else
  Status:=' is not greater than ';

 WriteLn(N1, Status, N2);
 ReadKey;{tunggu sampai ditekan sebuah tombol keyboard}
end.
```

Dalam Bahasa Pemrograman Free Pascal

```
uses CRT;

var N1, N2:Integer;
    Status:String;

begin
 Write('Please enter first number: ');
 ReadLn(N1);
 Write('Please enter second number: ');
 ReadLn(N2);

 if N1>N2 then
  Status:=' is greater than '
 else
  Status:=' is not greater than ';

 WriteLn(N1, Status, N2);
 ReadKey();{tunggu sampai ditekan sebuah tombol keyboard}
end.
```

Dalam Bahasa Pemrograman Delphi

```
uses System.SysUtils;

var N1, N2:Integer;
    Status:String;

begin
 Write('Please enter first number: ');
 ReadLn(N1);
 Write('Please enter second number: ');
 ReadLn(N2);

 if N1>N2 then
  Status:=' is greater than '
 else
  Status:=' is not greater than ';

 WriteLn(N1, Status, N2);
 ReadLn;{tunggu sampai ditekan sebuah tombol enter}
end.
```

Catatan Penting:

1. Hanya boleh ada 1 perintah **end.** di dalam bahasa pemrograman Delphi yang memiliki arti akhir dari program.
2. Perintah **ReadLn** dibuat sebelum **end.** agar tampilan program pada console tidak langsung hilang (agar hasil tampilan program dapat diamati) ketika program telah selesai dijalankan tetapi harus menekan tombol **Enter** dari keyboard untuk mengakhiri program.
3. Apapun yang diketikkan di dalam tanda { } akan diabaikan (tidak akan dijalankan) oleh bahasa pemrograman Delphi dan tidak akan dianggap sebagai kode program yang akan dijalankan.
4. Saat program selesai dijalankan maka hasil tampilan program akan hilang dan kembali ke Pascal / Delphi secara otomatis jika pada akhir program **Delphi** (sebelum perintah end.) tidak diletakkan **ReadLn;** (yang artinya komputer menunggu input sesuatu hingga ditekan enter) atau perintah **ReadKey();** pada akhir program **Free Pascal** atau **ReadKey;** untuk **Turbo Pascal** (yang artinya menunggu ditekan salah satu tombol keyboard tanpa harus menekan tombol enter). NB: Perintah yang dimaksud dapat dilihat pada 3 buah kode program di atas pada baris program yang ditandai kotak kecil berwarna merah.

Dapat diperhatikan bahwa ada beberapa perbedaan antara Pascal dan Delphi.

1. Pustaka Console (Text-Based / Command Prompt Based) untuk Pascal adalah CRT sementara pustaka Console untuk Dephi adalah SysUtils atau System.SysUtils.
2. Pemanggilan function atau fitur bawaan **Turbo Pascal** yang bertipe function tidak boleh menggunakan tanda ( ) pada akhir perintah atau function seperti halnya perintah ReadKey; pada program tersebut sementara pemanggilan function pada **Free Pascal** dan **Delphi** lebih fleksibel dimana penulisan tanda ( ) pada akhir function boleh dilakukan dan boleh tidak seperti perintah ReadKey; atau ReadKey();.

Bahasa Pemrograman Pascal Delphi

Berikut ini adalah pengelompokan tipe data dan variabel di dalam bahasa pemrograman Pascal dan Delphi.

1. Numerik bulat: Byte, SmallInt, Word, Integer, Cardinal
   NB: Tipe Integer pada Turbo Pascal memiliki jangkauan dan batasan memori sebesar 2 byte sementara tipe data Integer pada Free Pascal dan Delphi memiliki jangkauan dan batasan memori sebesar 4 byte). Sementara itu tipe data Cardinal juga tidak dimiliki oleh Turbo Pascal
2. Numerik pecahan: Real, Double, Extended
3. Karakter atau huruf: Char pada bahasa Pascal sementara AnsiChar & Char (Unicode) pada bahasa Delphi
4. String atau kalimat: String
   NB: Pada Bahasa pemrograman **Delphi**, tipe data string dibedakan menjadi **ShortString** atau **AnsiString** dan **String** yang dapat menyimpan aksara karakter **Unicode**, sementara tipe data string di Pascal hanya dapat menyimpan aksara dari kode ASCII)
5. Array yang dideklarasikan dengan menggunakan bantuan kata kunci **array** dan diakses dengan bantuan tanda [ ]

6. Pointer dan tipe data tingkat lanjutan lainnya

Berikut ini adalah sifat-sifat variabel di dalam bahasa pemrograman Pascal dan Delphi.

1. Setiap variabel yang akan digunakan dalam memrogram dengan Delphi wajib untuk dideklarasikan dengan tipe data yang spesifik terlebih dahulu sebelum pada posisi sebelum dimulainya keyword **begin** dari sebuah function ataupun program utama sebelum variabel tersebut dapat dikenal dan digunakan dalam program.
2. Mendeklarasikan variabel dalam bahasa pemrograman Delphi wajib dimulai dengan sebuah kata kunci **var**.
3. Jika variabel yang dideklarasikan tidak diberikan nilai awal maka Delphi secara otomatis akan memberikan nilai 0 untuk numerik.
4. Penulisan variabel ataupun pengenal lainnya seperti konstanta, tipe dan nama fungsi dalam huruf besar dan huruf kecil akan dianggap sebagai variable yang sama (case insensitive).

Berikut ini adalah contoh deklarasi variabel dalam bahasa pemrograman Delphi.

1. Mendeklarasikan variabel c1 dan c2 sebagai tipe data char

   ```
   var c1, c2 : char;
   ```

2. Mendeklarasikan variabel x sebagai salah satu tipe data integer

   ```
   var x : Integer;
   ```

3. Mendeklarasikan variabel y sebagai salah satu tipe data pecahan

   ```
   var y:Real;
   ```

4. Mendeklarasikan variable s sebagai salah satu tipe data string

   ```
   var s:ShortString;
   ```

Memberikan nilai kepada variable dalam bahasa pemrograman Delphi harus setelah sebuah fungsi atau program utama dimulai dengan kata kunci **begin**.

1. Memberikan sebuah huruf kepada variabel c1

   ```
   c1:='a';
   ```

2. Memberikan sebuah nilai numerik bulat kepada variabel x

   ```
   x:= 20;
   ```

3. Memberikan sebuah nilai numerik pecahan kepada variabel y

   ```
   y:= 10.5;
   ```

4. Memberikan sebuah nilai string kepada variable s

   ```
   s:='This is string';
   ```

Perintah atau simbol komentar adalah sebuah perintah atau simbol yang digunakan untuk memberitahukan bahasa pemrograman agar mengabaikan baris program tertentu untuk tidak dianggap sebagai program. Berikut ini adalah komentar dalam bahasa pemrograman Delphi.

```
program Project1;

{$APPTYPE CONSOLE}

{$R *.res}

uses System.SysUtils;
(*ini adalah sebuah komentar yang akan diabaikan oleh Delphi
atau tidak akan dianggap sebagai program oleh Delphi dimana
komentar ini ditandai atau dimulai dari tanda ( dan * tanpa
dipisah oleh spasi dan diakhiri dengan tanda * dan ) tanpa
dipisah oleh Jenis komentar ini berlaku lebih dari 1 baris
program. Komentar ini hanya sebagai keterangan atau pengingat
bagi programmer saja mengenai mengapa dia membuat kode
program tertentu di baris program tertentu sementara
komentar berikut ini yang menggunakan tanda //
hanya berlaku dalam 1 baris program tersebut saja dimana
sejak dibuatnya tanda // hingga ke akhir baris program akan
diabaikan oleh Delphi*)

var i:Integer;//deklarasikan variabel i sebagai bilangan bulat

begin
{ini juga merupakan komentar yang dapat ditulis lebih dari 1
baris kecuali tanda buka kurung kurawal diikuti oleh tanda $
dan sama-sama harus diakhiri dengan tanda }

 {$I-}//ini bukan komentar tetapi compiler directive di Delphi
end.
```

Berikut ini adalah beberapa fungsi konversi antara numerik dan string di dalam bahasa pemrograman Delphi. (NB: Operand dapat berupa variabel ataupun konstanta)

1. VariabelInteger=StrToInt(OperandString);
2. VariabelString=IntToStr(OperandTipeBIlanganBulat);
3. VarPecahan=StrToFloat(OperandString);
4. VarString=FloatToStr(OperandTipeBilanganPecahan);

Contoh cara menggunakan beberapa fungsi konversi tipe data. Perhatikan deklarasi variabel berikut ini.

```
var i:Integer;
    f:Real;
```

```
    s1, s2, s3:ShortString;
begin
 i:=10;
 f:=23.41;
 s1:='The value is';
 s2:='123.456';
 s3:='255';
 //mencoba konversi pada bagian ini
end.
```

1. Mengkonversi numerik menjadi string (kalimat)

```
s4 := s1 + IntToStr(i);
s4 := s1 + FloattoStr(f);
```

2. Mengkonversi string menjadi numerik pecahan (float)

```
f := 52.35 + StrToFloat(s2);
```

3. Mengkonversi string menjadi numerik bulat (integer)

```
i := 100 + StrToInt(s3);
```

Operator Aritmatika dan Fungsi Aritmatika Increment/Decrement

Operator	Kegunaan	Contoh
:=	Memberikan nilai dari operand kanan kepada operand kiri	a:=10;
+	Menghitung dan mengembalikan hasil penjumlahan antara 2 operand	b:=a+20;
-	Menghitung dan mengembalikan hasil selisih antara 2 operand	c:=b-30;
*	Menghitung dan mengembalikan hasil perkalian antara 2 operand	d:=c*2;
/	Menghitung dan mengembalikan hasil pembagian antara 2	e:=d/3;

	operand	
Div	Menghitung, membulatkan dan mengembalikan hasil pembagian antara 2 operand	f:=e div 5;
mod	Menghitung dan mengembalikan sisa pembagian antara 2 operand	f:=e mod 5;
Inc(opvar);	Menambahkan operand kiri dengan nilai 1	Inc(i);
Dec(opvar);	Mengurangi operand kiri dengan nilai 1	Dec(i);
Inc(ki, ka);	Menambahkan operand kiri dengan sebuah nilai dari operand kanan	Inc(i, 2);
Dec(ki, ka);	Mengurangi operand kiri dengan sebuah nilai dari operand kanan	Dec(i, 3);

Tabel 1 Operator Aritmatika

Operator pemeriksaan kondisi

Operator	Kegunaan	Contoh
=	Menghasilkan true jika kedua operand yang dibandingkan memiliki nilai yang sama dan menghasilkan false jika sebaliknya	if n%2==0:
<>	Menghasilkan true jika kedua operand yang dibandingkan memiliki nilai yang berbeda dan menghasilkan false jika sebaliknya	if a!=b:
<=	Menghasilkan true jika operand sebelah kiri lebih kecil atau sama dengan operand kanan dan menghasilkan false jika sebaliknya	if n<=10:
<	Menghasilkan true jika operand sebelah kiri lebih kecil operand kanan dan menghasilkan false jika sebaliknya	while n<=10:
>=	Menghasilkan true jika operand sebelah kiri lebih besar atau sama dengan operand kanan dan menghasilkan false jika sebaliknya	if y>=0:
>	Menghasilkan true jika operand sebelah kiri lebih besar operand kanan dan menghasilkan false jika sebaliknya	if x>0:

Tabel 2 Operator Pemeriksaan Kondisi

Operator logika

1. Operator And

Operand 1 (x)	Operand 2 (y)	x And y
False	False	False
False	True	False
True	False	False
True	True	True

Tabel 3 Operator And

2. Operator Or

Operand 1 (x)	Operand 2 (y)	x Or y
False	False	False
False	True	True
True	False	True
True	True	True

Tabel 4 Operator Or

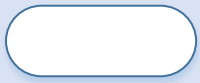
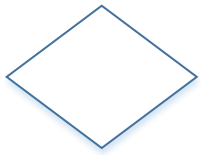
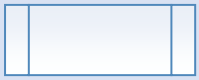
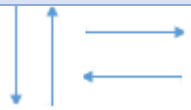
3. Operator Not

Operand	Not Operand
False	True
True	False

Tabel 5 Operator Not

Basic Flowchart

Flowchart Symbol	Arti

	Memulai atau mengakhiri program
	Input atau output program
	Proses komputasi
	Kondisi dan percabangan 2 (If – Then – Else)
	Sub proses komputer
	Alur program

Tabel 6 Basic Flowchart Simbol

Operator himpunan: in, not in

Prioritas operator dalam bahasa pemrograman Delphi.

Operator	Kategori
( )	Bukan operator tetapi lebih diutamakan dari operator aritmatika
*** / div mod**	Kali dan Bagi
+ -	Tambah dan Kurang
Shl Shr	Shift Bit Right dan Shift Bit Left
And	Bitwise AND
Or XOr	Bitwise OR and XOR
<= < > >=	Operator Perbandingan Besar dan Kecil
= <>	Operator Perbandingan Sama atau Tidak
:=	Operator Assignment
in, not in	Operator Himpunan
not, and, or	Operator Logika

Bab 2 – Input dan Output dalam Bahasa Pemrograman Pascal dan Delphi

Manfaat Pembelajaran:

1. Agar kebenaran hasil komputasi dari sebuah proses algoritma dalam sebuah program dapat dilihat maka cara mencetak ke layar console (output) sangat penting di dalam belajar pemrograman.
2. Agar dapat memastikan bahwa algoritma yang dirancang dan program yang dibuat dapat menghasilkan output yang benar maka algoritma dan program harus diuji dengan berbagai scenario dan nilai input yang berbeda sehingga menguasai cara memrogram dengan perintah input juga sangat penting di dalam belajar pemrograman.
3. Mampu membedakan input dan output dalam bahasa pemrograman Pascal / Delphi dan input dan output dalam bahasa pemrograman Pascal / Delphi dan agar mampu untuk memilih perintah input dan output yang sesuai untuk kondisi pemrograman tertentu.

Peralatan standar input utama bagi sebuah sistem komputer adalah **keyboard** meskipun sebuah sistem komputer juga memiliki peralatan input yang bukan standar input utama seperti microphone, camera, scanner dan lain-lain sehingga pembahasan input di dalam pemrograman adalah selalu mengutamakan penggunaan keyboard sebagai input bagi program komputer. Selain standar input, sebuah sistem komputer juga memiliki standar utama output yaitu **layar monitor** meskipun sebuah sistem komputer juga memiliki peralatan output yang bukan standar output utama seperti proyektor, speaker, printer dan lain-lain sehingga pembahasan output di dalam pemrograman adalah selalu mengutamakan penggunaan cetakan pada layar monitor baik cetakan yang berupa pesan, kalimat, keterangan ataupun cetakan yang berupa nilai numerik.

Jika pembaca berangkat dari ilmu matematika, maka untuk memahami pentingnya input dan output dari sebuah program dapat dimulai dengan permasalahan persamaan matematika dengan n buah variabel dimana nilai dari n-1 buah variabel telah diketahui seperti pada contoh soal persamaan matematika berikut ini.

1. Cari nilai z dari persamaan 5 + z = x – y jika x dan y diberikan sembarangan nilai (misalnya x=20 dan y=15)

   Penjelasan:

   - Jika nilai variabel z harus dicari, maka variabel z merupakan **output** program
   - Jika nilai variabel x dan y telah diketahui maka variabel x dan variabel y merupakan **input** program
   - Proses untuk mendapatkan nilai z dari nilai x dan y harus diselesaikan dengan mengimplementasikan algoritma yang tepat agar permasalahan persamaaan matematika tersebut dapat diprogram dan diselesaikan dengan salah satu bahasa pemrograman komputer yang ada

2. Cari nilai d dari persamaan 5c - 1= 2a / b jika a dan b diberikan sembarangan nilai (misalnya a=14 dan b=2)

   Penjelasan:

   - Jika nilai variabel c harus dicari, maka variabel c merupakan **output** program
   - Jika nilai variabel a dan b telah diketahui maka variabel a dan variabel b merupakan **input** program
   - Proses untuk mendapatkan nilai c dari nilai a dan b harus diselesaikan dengan mengimplementasikan algoritma yang tepat agar permasalahan persamaaan matematika tersebut dapat diprogram dan diselesaikan dengan salah satu bahasa pemrograman komputer yang ada

Dengan memahami nilai variabel yang diketahui (input program) dan dengan memahami nilai variabel yang dicari (output program) maka harapan penulis adalah bahwa pembaca dapat memahami konsep input dan output program sebagai nilai variabel yang diketahui dan nilai variabel yang dicari. Berikut ini pembahasannya

Ada beberapa perbedaan antara bahasa pemrograman C dan bahasa pemrograman C++. Bahasa pemrograman C++ adalah bahasa pemrograman sambungan dari bahasa pemrograman C dimana bahasa pemrograman C lebih bersifat procedural dan bahasa pemrograman C++

sudah mendukung pemrograman berorientasi objek (Object Oriented Programming / OOP). Sehingga bahasa pemrograman C lebih cocok digunakan pada level pengembangan kernel, OS, pemrograman hardware dan pengembangan low level lainnya yang lebih mendekati komunikasi antara OS dengan hardware. Sementara itu karena bahasa pemrograman C++ mendukung pemrograman berorientasi objek, maka pengembangan komponen perangkat lunak berbasis GUI seperti Form, Label, TextBox dan Button dan memungkinkan untuk memrogram software framework sehingga dengan adanya komponen atau framework perangkat lunak maka pengembangan perangkat lunak menjadi lebih mudah dengan C++ dan semua bahasa pemrograman keturunannya seperti Java, C#, JavaScript, PHP dan lain lain.

Salah satu perbedaan antara C dan C++ yang akan dibahas pada bagian ini adalah perintah input, output dan tipe data string dimana bahasa C menggunakan perintah input dan output seperti printf, scanf, puts dan gets dan memiliki tipe data null terminated string sedangkan bahasa C++ menggunakan perintah input dan output seperti cout, cin dan tipe data class string.

Bentuk Umum Perintah Output Delphi (Write dan WriteLn)

```
Write(obj1, obj2, ... , ObjN);
WriteLn(obj1, obj2, ... , ObjN);
```

Keterangan:

1. *Obj1, Obj2, … , ObjN* dapat berupa sebuah konstanta ataupun variabel yang bertipe numerik bulat (Byte, ShortInt, SmallInt, Word, Int, Cardinal, LongInt), numerik pecahan (Real, Double, Extended), huruf (Char) ataupun kalimat (ShortString, String dan UnicodeString).
2. *Obj1, Obj2, … , ObjN* adalah optional, artinya boleh mencetak hanya dengan 1 objek konstanta ataupun variable dalam 1 perintah Write, boleh juga mencetak lebih dari 1 objek dalam 1 perintah Write. Bahkan boleh melakukan WriteLn; tanpa objek apapun dengan tujuan hanya untuk memindahkan baris cetakan saja.

3. Perbedaan antara perintah Write dan WriteLn adalah bahwa posisi cetakan selanjutnya dari perintah Write akan bersambung di samping hasil cetakan sedangkan posisi cetakan selanjutnya dari perintah WriteLn akan bersambung di kiri bawah dari hasil cetakan.
4. Karakter Chr(13) + Chr(10) dapat digunakan sebagai salah satu objek cetakan dalam kedua perintah Write ini dengan tujuan untuk menghasilkan pindah baris di tengah cetakan dimana pada sistem hardware komputer, Chr(13) memiliki arti tombol Enter dan Chr(10) memiliki arti pindah baris.

Berikut ini adalah contoh pemanggilan perintah Write dan WriteLn.

```
1. Write('Hello World');
2. WriteLn('Hello World');
3. WriteLn('Hello World', Chr(13)+Chr(10) );
4. Write('Hello', Chr(13)+Chr(10) ,'World')
5. WriteLn(x, y, z);
6. WriteLn(x, ' ', y, ' ', z);
7. WriteLn(x, '   ', y, '   ', z);
8. WriteLn(x, '---', y, '---', z);
```

Bentuk Umum Perintah Input Delphi (ReadLn)

```
ReadLn(var);
```

Perintah input akan mengembalikan sebuah nilai dimasukkan oleh user melalui keyboard sesuai dengan tipe data dasar dari variable tersebut pada saat variable dideklarasikan.

Contoh:

```
Write('Please enter a number : ');
ReadLn(num);
```

Kegiatan Mandiri

1. Buat sebuah algoritma/pseudo code/flowchart (pilih salah satu) dan program dalam bahasa pemrograman Delphi untuk menetukan luas segitiga jika nilai alas dan tinggi segitiga dimasukkan ke program melalui keyboard.
2. Buat sebuah algoritma/pseudo code/flowchart (pilih salah satu) dan program dalam bahasa pemrograman Delphi untuk menetukan luas persegi panjang jika nilai panjang dan lebar persegi panjang dimasukkan ke program melalui keyboard.
3. Buat sebuah algoritma/pseudo code/flowchart (pilih salah satu) dan program dalam bahasa pemrograman Delphi untuk menetukan volume balok jika nilai panjang, lebar dan tinggi balok dimasukkan ke program melalui keyboard.
4. Buat sebuah algoritma/pseudo code/flowchart (pilih salah satu) dan program dalam bahasa pemrograman Delphi untuk menetukan kecepatan tempuh yang dapat dicapai jika nilai kecepatan tetap (contant) dan durasi (waktu) dimasukkan ke program melalui keyboard.
5. Buat sebuah algoritma/pseudo code/flowchart (pilih salah satu) dan program dalam bahasa pemrograman Delphi untuk menetukan keliling dan luas lingkaran jika nilai jari-jari atau diameter lingkaran dimasukkan ke program melalui keyboard. NB. Nilai konstanta pi di dalam bahasa pemrograman Delphi dapat langsung dipanggil dengan mengetikkan `pi`.

Bab 2 - Percabangan dan Exception Handling dalam Bahasa Pemrograman Delphi

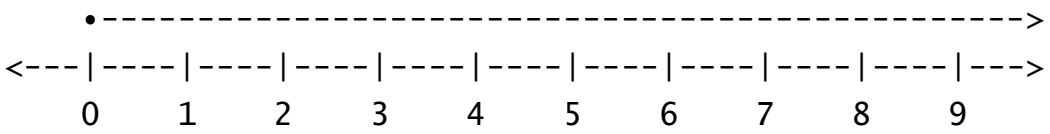
Manfaat Pembelajaran:

1. Solusi untuk permasalahan komputasi tidak terlepas dari kemampuan program komputer dalam mengambil keputusan atas kondisi logis tertentu sehingga mempelajari teknik percabangan sangat bermanfaat dalam menyelesaikan banyak permasalahan komputasi.
2. Pada saat program sedang berjalan (program runtime), ada beberapa kondisi yang dapat menyebabkan runtime error. Salah satu kondisi yang dapat menyebabkan runtime error adalah divide/division by zero, sehingga fitur exception handling dibutuhkan untuk mengatasi masalah-masalah runtime error seperti ini.

Bagi pembaca yang memiliki latar belakang ilmu matematika, maka penggunaan percabangan (yang membutuhkan kondisi dari hasil perbandingan logis) dapat memulai dari memahami garis bilangan dan diagram Venn.

Pemahaman dari penulisan garis bilangan sangat bermanfaat dalam penulisan kondisi percabangan if pada saat menuliskan kondisi seperti nilai>=60, nilai>=90 and nilai<=100 dan lain-lain. Berikut ini adalah beberapa contoh kondisi perbandingan yang digambarkan dengan garis bilangan beserta dengan penulisan notasi kondisi logis-nya.

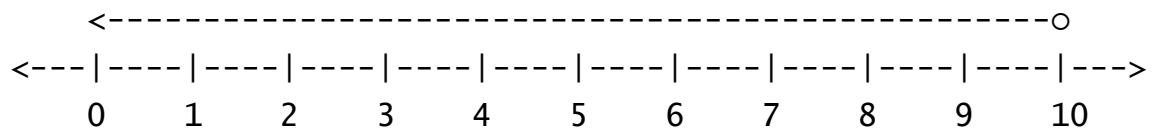
1. Garis Bilangan 1: kondisi **var>=0**



Penjelasan:

- Hanya memiliki 1 kondisi karena hanya ada 1 garis dan arah panah
- Menggunakan tanda > karena arah panah dari garis bilangan menunjuk ke arah kanan
- Menggunakan tanda = karena menggunakan bulatan penuh pada angka 0 yang artinya angka 0 diikutkan dalam kondisi himpunan

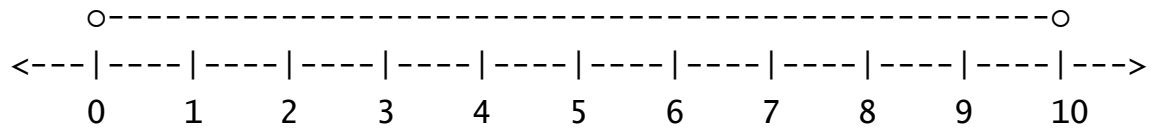
2. Garis bilangan 2: kondisi **var<10**



Penjelasan:

- Hanya memiliki 1 kondisi karena hanya ada 1 garis dan arah panah
- Menggunakan tanda < karena arah panah pada garis bilangan menunjuk ke arah kiri
- Tidak menggunakan tanda = karena menggunakan bulatan kosong pada angka 10

3. Garis bilangan 3: **var>0 and var<10**

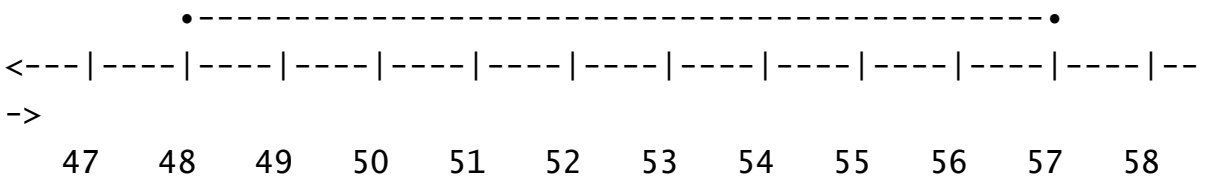


Penjelasan:

- Memiliki 2 kondisi karena ada 2 titik (angka) yang terlibat yaitu 0 dan 10
- Arah panah tertutup ke dalam karena kondisi variabelnya ada di dalam kedua angka yang terlibat (0 dan 10)
- Penulisan kondisi pertama (titik sebelah kiri yaitu angka 0) adalah **var>0**, menggunakan tanda > karena arah panah mengarah ke 10 (arah kanan) dimulai dari angka 0 dan tidak menggunakan tanda = karena bulatan kosong digunakan pada angka 0
- Penulisan kondisi kedua (titik sebelah kanan yaitu angka 10) adalah **var<10**, menggunakan tanda < karena arah panah mengarah ke 0 (arah kiri) dimulai dari 10 dan tidak menggunakan tanda = karena bulatan kosong digunakan pada angka 0
- Dalam notasi matematika memang dapat dituliskan dengan cara **0<var<10** tetapi dengan mengingat bahwa prosesor komputer hanya dalam melakukan 1 operasi dalam 1 clock prosesor (waktu prosesor yang mungkin dalam satuan milidetik) maka umumnya pemrograman komputer tidak akan mampu mengoperasikan 2 buah operator < pada notasi **0<var<10** dalam waktu yang bersamaan sehingga notasi **0<var<10** harus dipisahkan dulu menjadi **0<var** dan **var<10** agar bahasa pemrograman komputer dapat memahami bahwa notasi **0<var<10** harus dikerjakan secara terpisah menjadi **0<var** dan **var<10** atau **var>0** dan **var<10** kemudian kedua ekspresi kondisi tersebut harus ditulis

dengan cara menggabungkannya dengan operator **and** sehingga penulisan dalam bahasa pemrograman komputer akan menjadi **0<var and var<10** atau **var>0 and var<10** dimana operator **and** dipilih karena himpunan bilangan pada notasi **var>0** dan **var<10** memiliki **arah panah tertutup** (bersinggungan)

4. Garis bilangan 4: **var>=48 and var<=57**

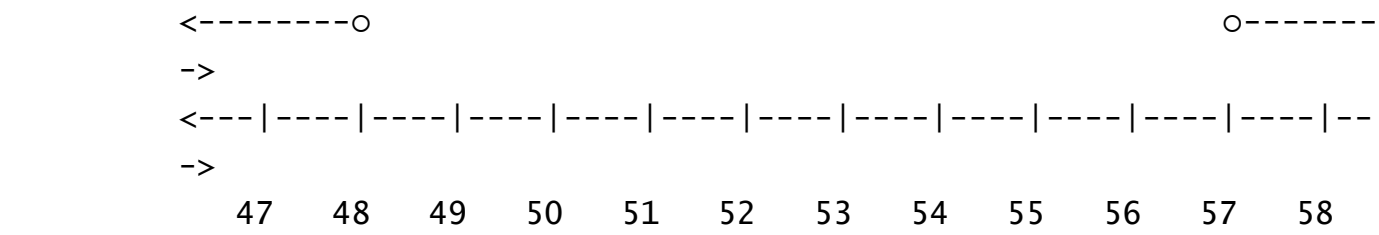


Penjelasan:

- Memiliki 2 kondisi karena ada 2 titik (bilangan) yang terlibat yaitu 48 dan 57
- Arah panah tertutup ke dalam karena kondisi variabelnya ada di dalam kedua bilangan yang terlibat (48 dan 57)
- Penulisan kondisi pertama (titik sebelah kiri yaitu 48) adalah **var>=48**, menggunakan tanda > karena arah panah mengarah ke 57 (arah kanan) dimulai dari 48 dan menggunakan tanda = karena bulatan penuh digunakan pada 48
- Penulisan kondisi kedua (titik sebelah kanan yaitu 57) adalah **var<=57**, menggunakan tanda < karena arah panah mengarah ke 48 (arah kiri) dimulai dari 57 dan tidak menggunakan = karena bulatan penuh digunakan pada 57
- Dalam notasi matematika memang dapat dituliskan dengan cara **48<=var<=57** tetapi dengan mengingat bahwa prosesor komputer hanya dalam melakukan 1 operasi dalam 1 clock prosesor (waktu prosesor yang mungkin dalam satuan milidetik) maka umumnya pemrograman komputer tidak akan mampu mengoperasikan 2 buah operator < pada notasi **48<=var<=57** dalam waktu yang bersamaan sehingga notasi **48<=var<=57** harus dipisahkan dulu menjadi **48<=var** dan **var<=57** agar bahasa pemrograman komputer dapat memahami bahwa notasi **48<=var<=57** harus dikerjakan secara terpisah menjadi **48<=var** dan **var<=57** atau **var>=48** dan **var<=57** kemudian kedua ekspresi kondisi tersebut harus ditulis dengan cara menggabungkannya dengan operator **and** sehingga penulisan dalam bahasa pemrograman komputer akan menjadi **48<=var and var<=57**

atau **var>=48 and var<=57** dimana operator **and** dipilih karena himpunan bilangan pada notasi **var>=48** dan **var<=57** memiliki **arah panah tertutup** (bersinggungan)

5. Garis bilangan 5: **var<48 and var>57**



Penjelasan:

- Memiliki 2 kondisi karena ada 2 titik (bilangan) yang terlibat yaitu 48 dan 57
- Arah panah terbuka ke luar karena kondisi variabelnya ada di dalam kedua bilangan yang terlibat (48 dan 57)
- Penulisan kondisi pertama (titik sebelah kiri yaitu 48) adalah **var<48**, menggunakan tanda < karena arah panah mengarah ke kiri dimulai dari 48 dan menggunakan tidak tanda = karena bulatan kosong digunakan pada 48
- Penulisan kondisi kedua (titik sebelah kanan yaitu 57) adalah **var>57**, menggunakan tanda > karena arah panah mengarah ke kanan dimulai dari 57 dan menggunakan = karena bulatan kosong digunakan pada 57
- Baik dalam notasi matematika ataupun notasi ekpresi program komputer memang dituliskan dengan cara **var<48 or var>57** karena himpunan bilangan yang terdapat di dalam kondisi **var<48** dan himpunan bilangan yang terdapat di dalam kondisi **var>57** tidak memiliki persinggungan (persamaan) sama sekali.

Selain itu, pemahaman akan diagram Venn juga sangat penting di dalam menuliskan kondisi pada perintah percabangan dalam pemrograman karena pemahaman akan diagram Venn akan sangat membantu dalam memilih operator manakah antara operator **and** dan operator **or** yang akan digunakan untuk menggabungkan di samping juga adanya operator **not** yang mungkin dibuuhkan dalam penggabungan notasi ekspresi kondisi logis.

Berikut ini adalah contoh soal diagram Venn yang membutuhkan pemahaman logis mengenai kapan menggunakan operator **and** (irisan ∩) **dan** operator or (Union/gabungan U)

Contoh soal:

Jika Himpunan A adalah beberapa merek laptop Acer, ASUS, Fujitsu, MSI, Lenovo, HP, Sony, Toshiba dan Himpunan B adalah beberapa merek mainboard komputer Gigabyte, MSI, Biostar, ASUS, Asrock, ECS, Foxconn maka buat notasi himpunan untuk kondisi berikut beserta hasil notasi himpunan tersebut.

1. Merek yang merupakan merek laptop dan merek mainboard komputer sekaligus
2. Merek yang merupakan merek laptop atau merek mainboard komputer
3. Merek yang merupakan merek laptop tapi bukan merupakan merek mainboard komputer
4. Merek yang merupakan merek mainboard komputer tetapi bukan merupakan merek laptop
5. Merek yang laptop dan mainboard lainnya yang bukan merupakan anggota himpunan A dan bukan merupakan anggota himpunan B

Jawaban

1. **A ∩ B** (**Kondisi A and Kondisi B**) yaitu ASUS dan MSI karena memenuhi kondisi himpunan A dan memenuhi kondisi himpunan B sekaligus.
2. **A U B** (**Kondisi A or Kondisi B**) yaitu Acer, ASUS, Fujitsu, MSI, Lenovo, HP, Sony, Toshiba dan Gigabyte, MSI, Biostar, ASUS, Asrock, ECS, Foxconn karena hanya perlu memenuhi salah satu dari kedua kondisi himpunan A atau kondisi himpunan B.
3. **A ∩ ~B (Kondisi A and not Kondisi B)** yaitu Acer, Fujitsu, Lenovo, HP, Sony dan Toshiba (ASUS dan MSI tidak termasuk karena keduanya juga terdapat di dalam kondisi B sementara permintaan soal adalah harus yang bukan merupakan anggota himpunan B) karena harus memenuhi Kondisi A yang tidak memenuhi kondisi B.
4. **B ∩ ~A (Kondisi B and not Kondisi A)** yaitu Gigabyte, BIOSTAR, Asrock dan ECS (ASUS dan MSI tidak termasuk karena keduanya juga terdapat di dalam kondisi A sementara permintaan soal adalah harus yang bukan merupakan anggota himpunan A) karena harus memenuhi Kondisi B yang tidak memenuhi Kondisi A.

5. **~A ∩ ~B atau ~(A U B)** yaitu merek-merek lainnya yang tidak terdapat di dalam kondisi A ataupun kondisi B misalnya Samsung, Hyundai, Zyrex, Axioo karena permintaan soal adalah tidak memenuhi kedua anggota himpunan dari himpunan A ataupun himpunan B.

Dengan adanya pembahasan mengenai garis bilangan, himpunan dan diagram Venn maka pembelajaran perintah percabangan if berikut ini menjadi lebih mudah dipahami karena perintah percabangan if menggunakan penulisan kondisi logis yang identik dengan penulisan notasi garis bilangan dan notasi diagram Venn. Berikut ini pembahasannya.

Ada 2 buah perintah percabangan dalam bahasa pemrograman Pascal / Delphi.

1. Percabangan if – then atau if – then – else
2. Percabangan case

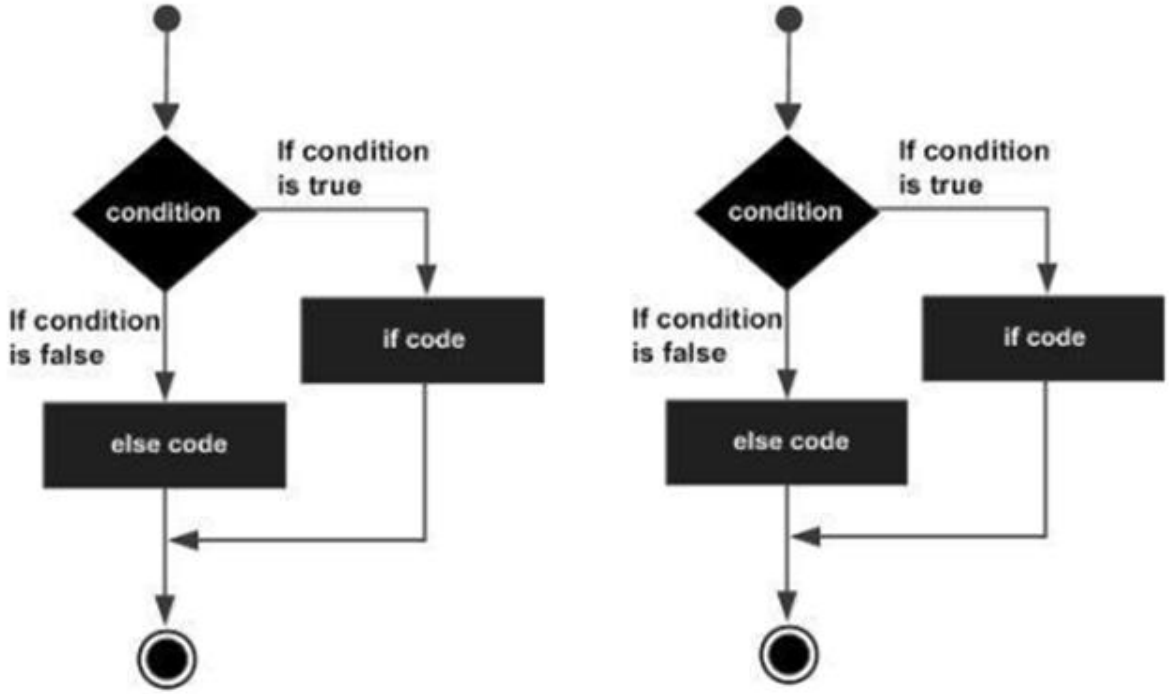
Bentuk umum perintah percabangan Delphi

1. if tanpa else

```
if condition then
 Statement;
```

2. if dengan else

```
if condition then
 Statement
else
 Statement;
```



3. if bersarang dengan perintah elif (else if)

```
if condition1 then
  statement
else if condition2 then
  statement
:
:
else if condition then
  statement
else
  statement;
```

4. perintah case (Hanya dapat digunakan untuk variabel bertipe numerik dan char dan hanya dapat menjalankan 1 perintah di dalam setiap case perbandingan nilai, agar sebuah case perbandingan nilai bisa menjalankan lebih dari 1 perintah maka dibutuhkan blok perintah **begin – end** dimana semua perintah yang akan dikerjakan oleh sebuah case perbandingan nilai diletakkan diantara perintah **begin** dan **end**.

```
case var of
 Nilai1: Perintah;
 Nilai2: Perintah;

 Nilai3: begin
          Perintah1;
          Perintah2;
          :
          PerintahN;
         end;
 else Perintah; {jika tdk ada nilai yg cocok dan else tidak wajib
ada}
end;
```

NB (Penting untuk diingat !!!): Jika kondisi yang kita butuhkan untuk diperiksa oleh if merupakan kondisi gabungan dari beberapa kondisi yang lebih kecil, maka diharuskan untuk menggunakan tanda ( ).

Berikut ini adalah contoh pemakaian if yang salah:

```
if kehadiran>11 and nilai>=60 then
 hasil:='lulus'
else
 hasil:='gagal';
```

Berikut ini adalah contoh pemakaian if yang benar:

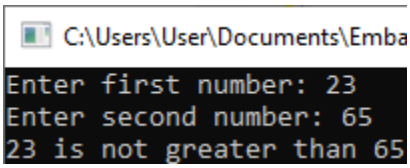
```
if (kehadiran>11) and (nilai>=60) then
 hasil:='lulus'
else
```

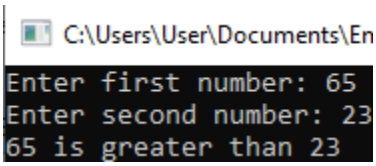
```
 hasil:='gagal';
```

Contoh penggunaan if tanpa else

```
program Project1;

{$APPTYPE CONSOLE}

{$R *.res}

uses System.SysUtils;

var N1, N2:Integer;
    status:ShortString;

begin
 Write('Enter first number: ');
 ReadLn(N1);
 Write('Enter second number: ');
 ReadLn(N2);

 status:='greater than ';
 if N1<=N2 then
  status:='not '+status;

 WriteLn(N1, ' is ', status, N2);
 ReadLn;//komputer menunggu kita metekan enter
end.//agar tampilan hasil program tidak langsung hilang
```

Hasil tampilan program (2x percobaan):





Contoh penggunaan if dengan else.

```
program Project1;

{$APPTYPE CONSOLE}

{$R *.res}

uses System.SysUtils;

var N1, N2:Integer;
    status:ShortString;

begin
 Write('Enter first number: ');
 ReadLn(N1);
 Write('Enter second number: ');
 ReadLn(N2);


 if N1>N2 then
  status:='greater than '
 else
  status:='not greater than ';

 WriteLn(N1, ' is ', status, N2);
 ReadLn;//komputer menunggu kita metekan enter
end.//agar tampilan hasil program tidak langsung hilang
```

Contoh penggunaan if bersarang.

```
program Project1;

{$APPTYPE CONSOLE}

{$R *.res}

uses System.SysUtils;

var N1, N2:Integer;
    status:ShortString;

begin
 Write('Enter first number: ');
 ReadLn(N1);
 Write('Enter second number: ');
 ReadLn(N2);


 if N1>N2 then
  status:='greater than '
 else if N1<N2 then
  status:='less than '
 else
  status:='is equal to ';

 WriteLn(N1, ' is ', status, N2);
 ReadLn;//komputer menunggu kita metekan enter
end.//agar tampilan hasil program tidak langsung hilang
```

Berikut ini contoh penggunaan if – then – else bersarang.

Perlu diperhatikan pada contoh berikut bahwa:

1. Setiap penggabungan kondisi dengan operator **And** atau **Or** maka kondisi yang akan digabungkan tersebut harus diketikkan dalam tanda ( ).
2. Penulisan kondisi logis pada perintah if dalam semua bahasa pemrograman harus dalam keadaan yang lengkap dengan bentuk **operand** *operator* **operand** *operator* **operand** *operator* **operand** dan seterusnya (operand dan operator harus ditulis berganti-gantian misalnya operand harus diikuti oleh operator, tidak boleh diikuti oleh operand, begitu juga dengan operator harus diikuti oleh operand, tidak boleh diikuti oleh operator). Sehingga contoh penulisan kondisi logis yang benar (Mark>60) and (Mark<=80) dan penulisan kondisi

Mark>60 and <=80 adalah penulisan yang salah karena operator and bertemu langsung dengan operator <= tanpa adanya operand diantara keduanya.

```
uses System.SysUtils;

var Grade:Char;
    Mark:Byte; (*byte adalah tipe data numerik
               dengan nilai jangkauan 0 s/d 255*)
begin
 Write('Please input your final exam mark: ');
 ReadLn(Mark);

 if (Mark>80) and (Mark<=100) then
  Grade:='A'
 else if (Mark>60) and (Mark<=80) then
  Grade:='B'
 else if (Mark>40) and (Mark<=60) then
  Grade:='C'
 else if (Mark>20) and (Mark<=40) then
  Grade:='D'
 else if (Mark>0) and (Mark<=20) then
  Grade:='E'
 else
  Grade:='F';

 WriteLn('You got ', Grade);
end.
```

Potongan program contoh penggunaan case (Misalkan Nomor adalah variabel bertipe short int dan Nama adalah variabel bertipe null terminated string).

```
case Nomor of
 1: Nama:='Senin';
 2: Nama:='Selasa';
 3: Nama:='Rabu';
 4: Nama:='Kamis';
 5: Nama:='Jumat';
 6: Nama:='Sabtu';
 7: Nama:='Minggu';
 else Nama:='Kiamat';
end;
```

Meskipun pada dasarnya penggunaan perintah **case** tersebut dapat digantikan dengan if – then – else seperti potongan contoh program berikut ini. Namun tetap saja perintah case memiliki beberapa kemudahan seperti tidak perlu mengetikkan variabel **Nomor** secara berulang-ulang, namun tetap harus diperhatikan bahwa hanya hanya perintah terakhir yang menggunakan

tanda ; karena dalam bahasa pemrograman Pascal / Delphi, tidak boleh ada tanda ; sebelum perintah else.

```
if Nomor=1 then Nama then='Senin'
else if Nomor=2 then Nama then='Selasa'
else if Nomor=3 then Nama then='Rabu'
else if Nomor=4 then Nama then='Kamis'
else if Nomor=5 then Nama then='Jumat'
else if Nomor=6 then Nama then='Sabtu'
else if Nomor=7 then Nama then='Minggu'
else Nama then='Kiamat';
```

Potongan contoh program berikut ini adalah contoh dimana untuk beberapa nilai yang mengerjakan perintah yang sama dapat dilakukan dengan menggunakan tanda koma.

```
case NomorBulan of
 1, 3, 5, 7, 8, 10, 12: JlhHari:=31;
 4, 6, 9, 11          : JlhHari:=30;
 2                    : JlhHari:=28;
end;
```

Exception Handling

Ada 2 jenis error dalam memrogram program komputer tapi hanya error jenis kedua yang dapat menyebabkan terjadinya exception.

1. Compilation Error (Including Semantic and Lexical Error)
   a. Error ini terjadi karena kesalahan semantic error, lexical dan kesalahan grammer dalam bahasa pemrograman.
   b. Error ini terjadi pada saat program masih sedang dikompilasi dan belum dijalankan pada mesin dan sistem operasi (OS) komputer.
2. Runtime Error

   Error yang terjadi pada saat program sudah berjalan pada mesin dan OS komputer. Error jenis ini yang menyebabkan terjadinya exception (exception thrown). Error ini terjadi karena perintah-perintah dan fungsi-fungsi bahasa pemrograman tidak mampu mengatasi error tersebut.

Berikut ini adalah cara mengatasi error tersebut dalam bahasa pemrograman Delphi.

1. Compilation Error – ditangani dengan cara mengikuti aturan bahasa pemrograman, termasuk tata cara penulisan semantic, lexical dan grammer.
2. Runtime Error – ditangani dengan cara menggunakan exception handling tetapi beberapa kondisi error tertentu juga dapat ditangani dengan menggunakan compiler directive {$}.

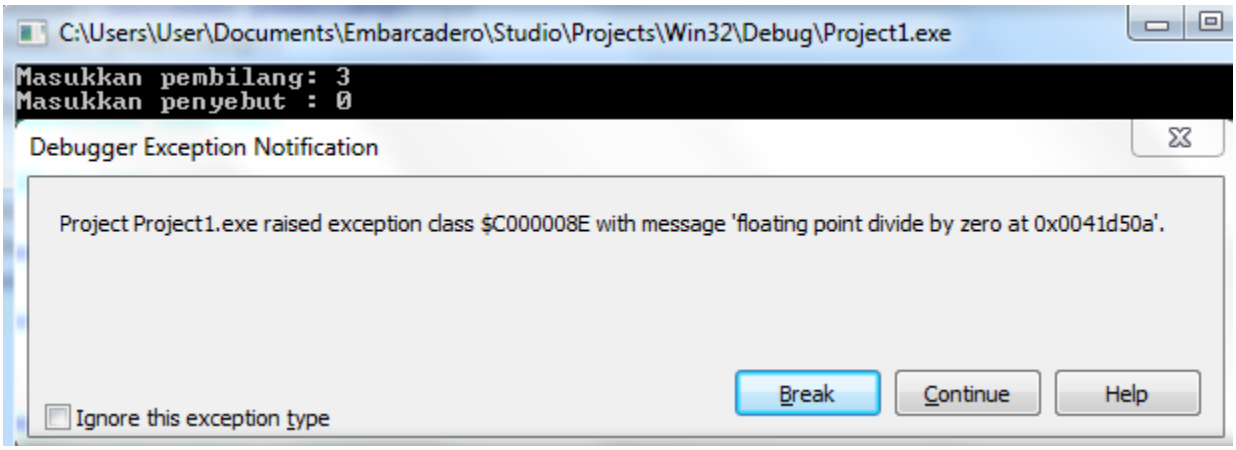
Berikut ini adalah bentuk penulisan perintah try dalam menangani exception (fitur ini tidak ada pada Turbo Pascal).

```
try
 //unpredictable statements
except
 on E: Exception do
  //statement to be executed if error occurs
  //E.ClassName and E.Message can be use as error message information
end;
```

Berikut ini adalah sebuah contoh program yang dapat menyebabkan exception.

```
uses System.SysUtils;

var pembilang, penyebut:SmallInt;
    pecahan:Real;

begin
 Write('Masukkan pembilang: ');
 ReadLn(pembilang);
 Write('Masukkan penyebut : ');
 ReadLn(penyebut);            //jika dimasukkan 0

 pecahan:=pembilang/penyebut; //diduga bisa error
 WriteLn('Hasilnya adalah ', pecahan);
 ReadLn;
end.
```

Berikut ini adalah tampilan program (jika dijalankan pada Delphi) yang menghasilkan (raise) exception ketika penyebut yang dimasukkan adalah 0.

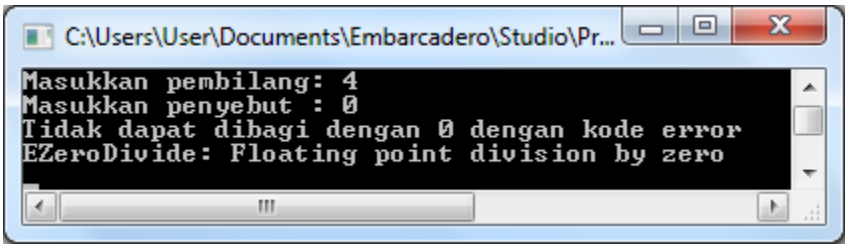


Dengan menggunakan bantuan perintah try – except – end seperti kode program berikut ini.

```
uses System.SysUtils;

var pembilang, penyebut:SmallInt;
    pecahan:Real;

begin
 Write('Masukkan pembilang: ');
 ReadLn(pembilang);
 Write('Masukkan penyebut : ');
 ReadLn(penyebut);              //jika dimasukkan 0

 try
  pecahan:=pembilang/penyebut; //diduga bisa error
  WriteLn('Hasilnya adalah ', pecahan);
 except
  on E: Exception do
   Writeln('Tidak dapat dibagi dengan 0 dengan kode error',
    Chr(13)+Chr(10), E.ClassName, ': ', E.Message);
 end;
 ReadLn;
end.
```

Maka meskipun kotak dialog error debugging tetap muncul, tetapi saat program tidak dijalankan dari RAD Studio, yaitu dengan cara melakukan double click pada file .exe dari hasil kompilasi yang executable maka kotak dialog error debugging tersebut tidak akan muncul

tetapi tampilan pesan error dari kode program yang kita buat yang akan muncul seperti dilihat pada tampilan output berikut.



Kegiatan Mandiri

1. Rancang sebuah algoritma/program Delphi untuk menentukan apakah sebuah angka yang dimasukkan dari keyboard antara 0 s/d 9 dimana jika input yang diberikan adalah 1 maka komputer akan menghasilkan **satu**.
2. Rancang sebuah algoritma/program Delphi untuk menentukan nama bulan dan jumlah hari pada bulan itu jika sebuah nomor bulan dimasukkan dari keyboard.
3. Perbaiki kegiatan mandiri nomor 1 dimana jika angka yang dimasukkan dari keyboard tidak di antara 0 s/d 9 maka cetak sebuah pesan bahwa hanya angka 0 s/d 9 yang boleh dimasukkan ke program.
4. Perbaiki kegiatan mandiri nomor 2 dimana jika nomor bulan yang dimasukkan tidak berada diantara 1 s/d 12 maka program akan mencetak pesan bahwa nomor yang dimasukkan tidak benar.
5. Rancang sebuah algoritma/program Delphi untuk menentukan nilai huruf jika dimasukkan nilai angka berdasarkan aturan penilaian berikut ini.

   Grade A+: 95<Mark<=100

   Grade A: 90<Mark<=95

   Grade A-: 85<Mark<=90

   Grade B+: 80<Mark<=85

   Grade B: 75<Mark<=80

Grade B-: 70<Mark<=75

Grade C+: 60<Mark<=65

Grade C: 55<Mark<=60

Grade C-: 50<Mark<=55

Grade D: 35<Mark<=50

Grade E: 0<=Mark<=35

Bab 3 – Perulangan (Looping) dengan Jumlah Perulangan yang belum Diketahui

Manfaat Pembelajaran:

Menggunakan kode program yang sama persis dan berulang berulang-ulang, membuat program komputer menjadi tidak efisien sehingga dibutuhkan perintah perulangan untuk menyelesaikan permasalahan komputasi tersebut.

Salah satu kelebihan program komputer yang sangat ampuh dalam menyelesaikan permasalahan komputasi adalah perintah perulangan. Menggunakan kode program yang sama persis dan berulang berulang-ulang, membuat program komputer menjadi tidak efisien dan boros. Menggunakan perulangan untuk menyelesaikan permasalahan ini adalah solusi yang paling tepat. Perulangan memungkinkan untuk mengulangi proses komputasi yang sama persis tetapi proses komputasi dilakukan terhadap data-data yang berbeda sehingga program komputer dapat digunakan untuk mengolah data dalam jumlah yang banyak meskipun sebenarnya program komputer memroses data-data tersebut secara satu per satu dan bukan sekaligus.

Penulisan perulangan dengan flowchart dapat dilakukan hanya dengan menggunakan aliran program (flow) dengan arah ke atas sedangkan algoritma dan pseudo code dapat dilakukan dengan 2 buah cara sebagai berikut.

Cara	Algoritma	Pseudo code
1	Atur_Nilai_Awal_Variabel Perulangan Ulang: KumpulanPerintahProgram Perubahan_Variabel_Atau_Kondisi Jika kondisi Maka Lompat ke Ulang	Set Variable Starting Value Ulang: KumpulanPerintahProgram Perubahan_Variabel_Atau_Kondisi IF kondisi THEN GOTO ke Ulang
2	Ulangi var dari NilaiAwal s/d NilaiAkhir Perubahan konstanta KumpulanPerintahProgram Akhir Perulangan	FOR var = NilaiAwal TO NilaiAkhir STEP konstanta KumpulanPerintahProgram END FOR

Ada 3 perintah perulangan di dalam bahasa pemrograman Delphi yaitu perintah while, repeat until dan perintah for.

- Perintah perulangan **while** memiliki karakteristik memeriksa kondisi perulangan terlebih dahulu sebelum mengulang perintah-perintah yang diberikan
- Perintah perulangan **repeat** - **until** memiliki karakteristik mengulang perintah-perintah yang terdapat di dalamnya minimal satu kali (1x) dan pemeriksaan kondisi perulangan dilakukan pada akhir perulangan yaitu pada perintah until.
- Baik perintah perulangan **while** ataupun perintah perulangan **repeat** – **until** digunakan ketika programmer tidak mengetahui atau tidak dapat memprediksi berapa kali perulangan yang dibutuhkan di dalam program tersebut.
- Perintah perulangan **while** mengulang perintah program yang ada di dalamnya selama kondisi yang diberikan kepada perintah **while** menghasilkan nilai true sedangkan perintah **repeat - until** mengulang perintah program yang ada di dalamnya hingga kondisi yang dibuat pada perintah **until** masih belum terpenuhi (bernilai **false**), jadi selama kondisi yang diberikan kepada perintah **until** menghasilkan **false**, maka perulangan **repeat – until** akan dihentikan.
- perintah perulangan **for** memiliki karakteristik hanya mengulang perinta-perintah yang diberikan menggunakan kondisi numerik dengan nilai awal dan nilai akhir variabel numerik tersebut sehingga perulangan for digunakan hanya apabila jumlah perulangan yang dibutuhkan oleh programmer sudah diketahui atau sudah bisa diprediksi oleh programmer. (akan dibahas pada bab selanjutnya)

Berikut ini adalah 2 bentuk penulisan perintah perulangan while dimana perintah while hanya bias digunakan untuk mengulang 1 perintah program saja dan block **begin – end** digunakan untuk mengulang lebih dari 1 perintah program tetapi hanya yang terdapat di dalam blok begin dan end saja.

```
while kondisi do
 PerintahProgram;


while kondisi do begin
 KumpulanPerintahProgram;
end;
```

NB:

Menggunakan margin spasi sebanyak 1 spasi atau lebih untuk penulisan perintah program di dalam perulangan dalam bahasa pemrograman Delphi tidak memiliki pengaruh apapun terhadap hasil program.

Contoh-contoh penggunaan perintah program while.

Contoh penggunaan kondisi dalam perintah while.

```
uses System.SysUtils;

var repeating:Boolean;
    choice:ShortInt;

begin
 repeating:=True;
 while repeating do begin
  Write('Please input 1 to display this message again: ');
  ReadLn(choice);
  if choice=1 then
   repeating:=True
  else
   repeating:=False;
 end;
 ReadLn;
end.
```

Contoh Infinite loop.

```
uses System.SysUtils;

begin
 while true do
  WriteLn('This message can''t be stop');
end.
```

Karena tanda kutip satu ' di dalam bahasa pemrograman Delphi memiliki arti memulai dan mengakhiri sebuah konstanta string maka pemberian sebuah tanda kutip satu ' sebagai tipe data string harus diketikkan sebanyak dua kali seperti pada kata **can''t**.

Catatan Penting: Tidak seperti bahasa pemrograman C/C++, Java, C#, JavaScript, PHP dan Python, bahasa pemrograman Delphi tidak memiliki perintah *break* untuk menghentikan perulangan secara paksa ataupun perintah *continue* untuk memaksa alur perulangan program diulang kembali dari awal.

Contoh penggunaan perintah while yang menyebabkan tidak ada satupun perintah perulangan yang akan diulang pada contoh penggunaan perintah while berlapis (nested while) berikut ini.

```
uses System.SysUtils;

var choice:Char;
    num, total:ShortInt;

begin
 total:=0;
 while choice='Y' do begin
  Write('Enter a number: ');
  ReadLn(num);
  total:=total+num;  //atau Inc(total, num);
  while (choice<>'Y') and (choice<>'N') do begin
   Write('Next number (Y/N) ? ');
   ReadLn(choice);
  end;
 end;
end.
```

Perlu diperhatikan bahwa while akan mengulang semua perintah yang berada di dalam blok **begin** - **end** jika kondisi choice='Y' menghasilkan true dan pemeriksaan kondisi variabel choice oleh while pada bagian yang diberikan kotak pada contoh di bawah ini dilakukan pada saat variabel choice sama sekali belum diberikan nilai awal, sehingga nilai variabel choice bisa saja bernilai 0 dan ini menyebabkan kondisi choice='Y' menjadi **false** yang mengakibatkan perintah while tidak akan mengulang perintah-perintah yang ada di dalam block **begin** - **end** dan menyebabkan program tersebut tidak akan menghasilkan tampilan apapun.

Untuk mengatasi hal ini maka dibutuhkan pemberian nilai awal variabel choice sebelum memulai perulangan while seperti hasil modifikasi yang dilakukan kepada program tersebut seperti kode program di bawah ini.

```
uses System.SysUtils;

var choice:Char;
    num, total:ShortInt;

begin
 total:=0;
 choice:='Y';
 while choice='Y' do begin
  Write('Enter a number: ');
  ReadLn(num);
  total:=total+num;  //atau Inc(total, num);
  choice:='E';
  while (choice<>'Y') and (choice<>'N') do begin
   Write('Next number (Y/N) ? ');
   ReadLn(choice);
  end;
 end;
end.
```

Setelah membahas perulangan while, berikut ini adalah bentuk penulisan perintah perulangan **repeat** – **until**.

```
repeat
 KumpulanPerintahProgram;
until kondisi;
```

Untuk mengatasi pemberian nilai awal variabel choice di atas, maka perintah repeat – until dapat dijadikan solusi, dimana perintah perulangan ini akan mengulang perintah yang terdapat di dalamnya dulu tanpa memeriksa kondisi terlebih dahulu. Berikut ini adalah contoh program dari perbaikan program perulangan nested while sebelumnya.

Penjelasan, perhatikan bahwa kondisi perintah **while** dan kondisi perintah perulangan **repeat – until** dalam mengulang adalah berkebalikan dimana **while** mengulang selama kondisinya **true** sedangkan **repeat – until** hanya akan mengulang jika kondisinya **false**. Sehingga hal ini menyebabkan konversi dari perintah perulangan **while** menjadi perintah **repeat – until** membutuhkan pembalikan kondisi: `(choice<>'Y') and (choice<>'N')` harus dibalikkan dengan operator not seperti `(not (choice<>'Y') and (choice<>'N') )` atau menjadi `(choice='Y') or`

`(choice='N')` dimana operator **and** akan menjadi operator **or** jika dibalikkan dengan operator **not** dan tanda <> akan menjadi = jika dibalikkan dengan operator **not**.

Kegiatan Mandiri.

1. Buat sebuah program untuk menentukan nama hari jika sebuah nomor hari dimasukkan dengan keyboard. Ketika nama hari sudah ditampilkan, program akan menanyakan apakah akan mencoba lagi atau tidak. Contoh tampilan input dan output boleh mengikuti tampilan di bawah ini dengan tampilan bergaris bawah adalah nomor hari yang diinput dari keyboard dan hasil output nama hari

```
Masukkan nomor hari: 4
Hari Kamis
```

2. Buat sebuah program untuk memasukkan sebuah bilangan secara berulang dan dan hitung total dari semua bilangan yang dimasukkan tersebut. Jika sebuah bilangan bernilai 0 dimasukkan maka perulangan akan dihentikan dan program akan menghitung dan menampilkan total dan rata-rata dari semua bilangan yang dimasukkan tersebut.

3. Buat sebuah program untuk memasukkan sebuah bilangan secara berulang-ulang kemudian cari nilai minimum dan maksimumnya. Ketika sebuah bilangan bernilai 0 dimasukkan maka perulangan harus dihentikan dan program harus menampilkan nilai minimum dan nilai maksimum.

Bab 4 – Perulangan dengan Jumlah Perulangan yang sudah Diketahui

Manfaat Pembelajaran:

Menggunakan kode program yang sama persis dan berulang-ulang dalam jumlah perulangan yang sudah dapat dipastikan membuat program komputer menjadi tidak efisien sehingga dibutuhkan perintah perulangan yang memiliki batasan nilai awal dan nilai akhir untuk menyelesaikan permasalahan komputasi tersebut.

Pada umumnya program komputer bekerja dengan operasi-operasi aritmatika dan beberapa algoritma dan program perlu melakukan operasi yang sama secara berulang-ulang dengan jumlah perulangan yang sudah dapat dipastikan terhadap data yang berbeda untuk menghasilkan sebuah output tertentu. Perintah perulangan for adalah sebuah perintah perulangan yang sangat cocok untuk menyelesaikan permasalahan tersebut. Umumnya perintah perulangan for bekerja dengan menggunakan sebuah variabel yang harus dimulai dari sebuah nilai awal proses perulangan akan membuat nilai variabel tersebut bertambah ataupun berkurang hingga variable tersebut mencapai nilai akhir sebagai ketentuan (kondisi) bagi perulangan for untuk berhenti.

Bagi pembaca yang memiliki dasar ilmu matematika maka konsep perulangan dengan jumlah perulangan yang sudah diketahui dapat diidentikkan dengan perhitungan Sigma seperti dua soal berikut ini.

1. ∑ (3i-1) untuk I dari 1 s/d 10 yang memiliki proses berulang sebagai berikut

Pada aat nilai I bernilai	Maka 3*i-1 bernilai	Maka sigma / total akan bernilai
		0 (nilai netral awal)
1	3 * 1 – 1 = 2	0 + nilai kolom 2 = 0 + 2 = 2
2	3 * 2 - 1 = 5	2 + nilai kolom 2 = 2 + 5 = 7

3	3 * 3 - 1 = 8	7 + nilai kolom 2 = 7 + 8 = 15
4	3 * 4 - 1 = 11	15 + nilai kolom 2 = 15 + 11 = 26
5	3 * 5 - 1 = 14	26 + nilai kolom 2 = 26 + 14 = 40
6	3 * 6 - 1 = 17	40 + nilai kolom 2 = 40 + 17 = 57
7	3 * 7 - 1 = 20	57 + nilai kolom 2 = 57 + 20 = 77
8	3 * 8 - 1 = 23	67 + nilai kolom 2 = 67 + 23 = 100
9	3 * 9 - 1 = 26	100 + nilai kolom 2 = 100 + 26 = 126
10	3 * 10 - 1 = 29	126 + nilai kolom 2 = 126 + 29 = 155

2. $\sum (2i^2)$ untuk I dari 1 s/d 5

Pada aat nilai I bernilai	Maka 2*i^2 bernilai	Maka sigma / total akan bernilai
		0 (nilai netral awal)
1	2*1^2 = 2*1 = 2	0 + nilai kolom 2 = 0 + 2 = 2
2	2*2^2 = 2*4 = 8	2 + nilai kolom 2 = 2 + 8 = 10
3	2*3^2 = 2*9 = 18	10 + nilai kolom 2 = 10 + 18 = 28
4	2*4^2 = 2*16 = 32	28 + nilai kolom 2 = 28 + 32 = 60
5	2*5^2 = 2*25 = 50	60 + nilai kolom 2 = 60 + 50 = 110

Dengan ada pembahasan kasus statistik (seperti total, rata-rata, nilai maksimum dan nilai minimum) dan sigma, maka kita akan memahami bahwa pembahasan tersebut sangat identic dengan perintah perulangan for yang menggunakan nilai awal dan kondisi nilai akhir dalam perulangannya sama seperti cara menghitung sigma dalam matematika.

Berikut ini adalah beberapa bentuk umum perulangan for dengan jumlah perulangan yang sudah diketahui dalam bahasa pemrograman Pascal / Delphi.

```
1. for var := NilaiAwal to NilaiAkhir do
```

Mengulangi satu perintah program yang diketikkan setelah perintah **do** dari sejak **var** bernilai **NiliAwal**, kemudian **var** akan terus *bertambah* 1 secara otomatis untuk tiap perulangan hingga **var** akan bernilai **NilaiAkhir** (perulangan akan dihentikan ketika **var**=**NilaiAkhir**). Contoh

```
for i := 2 to 4 do  //output program akan menghasilkan
 Write(i, ' ');  //nilai variabel i yang tercetak berubah dari 2 3 4
```

2. `for var := NilaiAwal downto NilaiAkhir do`

Mengulangi satu perintah program yang diketikkan setelah perintah **do** dari sejak **var** bernilai **NiliAwal**, kemudian **var** akan terus *berkurang* 1 secara otomatis untuk tiap perulangan hingga **var** akan bernilai **NilaiAkhir** (perulangan akan dihentikan ketika **var**=**NilaiAkhir**). Contoh

```
for i := 4 downto 2 do  //output program akan menghasilkan
 Write(i, ' ');  //nilai variabel i yang tercetak berubah dari 4 3 2
```

Pengaturan cetakan

Pascal / Delphi memiliki kemampuan dalam merapikan cetakan dengan perintah Write dan WriteLn dengan meletakkan :x untuk objek bertipe bilangan bulat dan meletakkan :x:y untuk objek bertipe data bilangan pecahan setelah mengetikkan variabel yang akan dicetak. :x artinya jumlah tempat (digit/huruf) yang harus disediakan untuk mencetak variabel tersebut sedangkan :y artinya hanya jumlah digit pecahan sebesar y yang harus ditampilkan. x dan y harus merupakan bilangan bulat positif dimana disarankan agar nilai x (jumlah tempat yang diberikan untuk pencetakan) selalu lebih besar atau sama dengan jumlah digit numerik bulat yang dicetak. Berikut ini contoh program cetakan yang belum diatur dengan rapi dengan bantuan :x dan :x:y beserta dengan hasil tampilan program.

```
var i, j:Integer;
    f:Real;
begin
 for i:=0 to 20 do begin
  j:=i*10;
  f:=i/3;
  WriteLn(i, ' ', j, ' ', f);
 end;
end.
```

```
 0   0  0.00000000000000E+0000
 1  10  3.33333333333333E-0001
 2  20  6.66666666666667E-0001
 3  30  1.00000000000000E+0000
 4  40  1.33333333333333E+0000
 5  50  1.66666666666667E+0000
 6  60  2.00000000000000E+0000
 7  70  2.33333333333333E+0000
 8  80  2.66666666666667E+0000
 9  90  3.00000000000000E+0000
10 100  3.33333333333333E+0000
11 110  3.66666666666667E+0000
12 120  4.00000000000000E+0000
13 130  4.33333333333333E+0000
14 140  4.66666666666667E+0000
15 150  5.00000000000000E+0000
16 160  5.33333333333333E+0000
17 170  5.66666666666667E+0000
18 180  6.00000000000000E+0000
19 190  6.33333333333333E+0000
20 200  6.66666666666667E+0000
```

Berikut ini contoh program yang sudah diperbaiki dengan mengatur cetakan dengan bantuan :x dan :x:y.

```
var i, j:Integer;
    f:Real;
begin
 for i:=0 to 20 do begin
  j:=i*10;
  f:=i/3;
  WriteLn(i:3, j:4, f:6:2);
 end;
end.
```

```
  0   0  0.00
  1  10  0.33
  2  20  0.67
  3  30  1.00
  4  40  1.33
  5  50  1.67
  6  60  2.00
  7  70  2.33
  8  80  2.67
  9  90  3.00
 10 100  3.33
 11 110  3.67
 12 120  4.00
 13 130  4.33
 14 140  4.67
 15 150  5.00
 16 160  5.33
 17 170  5.67
 18 180  6.00
 19 190  6.33
 20 200  6.67
```

Perhatikan bahwa pada hasil cetakan pertama, hasil cetakan variabel j tidak rapi dan hasil cetakan variabel f menjadi tampilan exponensial. Sementara pada hasil cetakan kedua:

- i:3 artinya:

  jika i bernilai 0 s/d 9 (hanya 1 digit) maka tetap diberikan 3 tempat untuk mencetak dimana sisa tempat 2 digit akan diisi oleh spasi (jumlah digit yang diberikan – jumlah digit i = 3-1 = 2 digit spasi)

  Jika I bernilai 10 s/d 99 (2 digit numerik) maka tetap diberikan 3 tempat untuk mencetak dimana sisa tempat 1 digit akan diisi oleh spasi (jumlah digit yang diberikan – jumlah digit I = 3-2 = 1 digit spasi)

- j:4 artinya:

jika j bernilai 0 s/d 9 (hanya 1 digit) maka tetap diberikan 4 tempat untuk mencetak dimana sisa tempat 3 digit akan diisi oleh spasi (jumlah digit yang diberikan – jumlah digit j = 4-1 = 3 digit spasi)
Jika j bernilai 10 s/d 99 (2 digit numerik) maka tetap diberikan 4 tempat untuk mencetak dimana sisa tempat 2 digit akan diisi oleh spasi (jumlah digit yang diberikan – jumlah digit j = 4-2 = 2 digit spasi)
Jika j bernilai 100 s/d 999 (3 digit numerik) maka tetap diberikan 4 tempat untuk mencetak dimana sisa tempat 1 digit akan diisi oleh spasi (jumlah digit yang diberikan – jumlah digit j = 4-1 = 1 digit spasi)

- f:6:2 artinya:
  Sediakan 6 digit untuk mencetak f dimana diantara 6 digit tersebut, 2 digit harus digunakan untuk jumlah digit pecahan yang harus dicetak dan 1 digit harus digunakan untuk mencetak tanda pemisah pecahan (tanda titik) sehingga menyisakan 3 digit untuk digit di depan pemisah pecahan. Beberapa digit dari 3 digit untuk pemisah pecahan tersebut juga akan diisi oleh spasi jika jumlah digit non pecahan dari nilai f lebih kecil dari 3 digit.

Kegiatan Mandiri - perulangan biasa

1. Buat sebuah program untuk mencetak deret 2 5 8 11 14 secara berurutan dengan bantuan perintah perulangan
2. Buat sebuah program untuk mencetak deret -2 5 -8 11 -14 secara berurutan dengan bantuan perintah perulangan
3. Buat sebuah program untuk mencetak deret +2 -+5 -8 +11 -14 secara berurutan dengan bantuan perintah perulangan
4. Buat program untuk menghitung hasil deret +2 -+5 -8 +11 -14 secara berurutan dengan bantuan perintah perulangan
5. Buat sebuah program untuk menampilkan bilangan fibonacci berikut ini 1 1 2 3 5 8 13 21
6. Buat sebuah program untuk memasukkan sebuah bilangan berulang-ulang sambil dihitung total dari semua nilai yang dimasukkan tersebut. Apabila bilangan yang dimasukkan adalah 0 maka perulangan harus dihentikan dan program harus menghasilkan total dan rata-rata dari nilai-nilai tersebut

7. Sama seperti nomor 6, tetapi program harus dapat memperoleh nilai minimum dan maksimum

Kegiatan Mandiri – sistem bilangan matematika

8. Buat sebuah program untuk menampilkan bilangan genap antara 1 s/d n dimana n harus dimasukkan dari keyboard.
9. Buat sebuah program untuk menampilkan bilangan ganjil antara 1 s/d n dimana n harus dimasukkan dari keyboard.

Untuk mengerjakan kegiatan mandiri nomor 10 s/d 14, teknik yang digunakan untuk mencari, mendapatkan atau menentukan apa yang diminta pada soal nomor 10 s/d 14 tersebut, membutuhkan teknik brute and force, yaitu dengan membuat sebuah perulangan yang mengulang dari nilai 1 s/d bilangan yang masukkan dari keyboard kemudian dilakukan percobaan membagi bilangan dengan teknik bruce force seperti ini (Misalkan nilai yang dimasukkan adalah 8).

```
Jika 8 dibagi 1 bersisa 0 Maka ((jalankan perintah tertentu))
Jika 8 dibagi 2 bersisa 0 Maka ((jalankan perintah tertentu))
Jika 8 dibagi 3 bersisa 0 Maka ((jalankan perintah tertentu))
Jika 8 dibagi 4 bersisa 0 Maka ((jalankan perintah tertentu))
Jika 8 dibagi 5 bersisa 0 Maka ((jalankan perintah tertentu))
Jika 8 dibagi 6 bersisa 0 Maka ((jalankan perintah tertentu))
Jika 8 dibagi 7 bersisa 0 Maka ((jalankan perintah tertentu))
Jika 8 dibagi 8 bersisa 0 Maka ((jalankan perintah tertentu))
```

Kegiatan Mandiri – sistem bilangan matematika dan teknik brute and force

10. Buat sebuah program untuk mencari faktor perkalian dari sebuah bilangan yang dimasukkan dari keyboard.

    Contoh tampilan output:

    ```
    Masukkan sebuah bilangan: 12
    Faktor perkaliannya adalah 1 2 3 4 6 12
    ```

11. Buat sebuah program untuk memperoleh bilangan prima antara 1 s/d n dimana n dimasukkan dari keyboard.

    Contoh tampilan output:

    ```
    Masukkan batas maksimum bilangan prima: 12
    2 3 5 7 11
    ```

12. Buat sebuah program untuk menentukan nilai Kelipatan Persekutuan Terkecil (KPK) dari sebuah bilangan yang dimasukkan dari keyboard.
13. Buat sebuah program untuk menentukan nilai Faktor Persekutuan Terbesar (FPB) dari sebuah bilangan yang dimasukkan dari keyboard.
14. Buat sebuah program untuk memasukkan sebuah pembilang dan penyebut dari sebuah pecahan kemudian lakukan penyederhanaan pada pecahan tersebut dan tampilkan.

Bab 5 –Perulangan Bersarang (Nested Loop)

Manfaat Pembelajaran:

Pada beberapa permasalahan pemrograman, sebuah perulangan tidak cukup untuk menyelesaikan masalah tersebut. Hal ini disebabkan karena seringnya ditemukan kasus dimana bagian yang berulang dalam program tersebut juga ternyata digunakan berulang-ulang dengan pola yang sama tetapi hasil program yang mungkin bisa berbeda.

NB:

Salah satu permasalahan yang membutuhkan nested loop adalah pengurutan, namun bab ini hanya berupa latihan karena permasalahan pengurutan yang membutuhkan nested loop dalam penyelesaiannya baru akan dibahas pada topik array.

Contoh kasus penyusunan pola dengan perulangan bersarang.

```
11111
22222
33333
44444
55555
```

Dapat dibuat dengan bahasa pemrograman Delphi seperti berikut ini (untuk versi Pascal harus mengganti **uses System.SysUtils;** dengan **uses CRT;**.

```
uses System.SysUtils;

var brs, klm:Integer;

begin
 for brs:=1 to 5 do begin
  for klm:=1 to 5 do
   Write(brs);
  WriteLn;
 end;
 ReadLn;
end.
```

Kegiatan Mandiri - perulangan bersarang/nested loop: Selesaikan pola cetakan berikut ini dengan menggunakan perulangan yang berlapis.

1.

```
11111
22222
33333
44444
55555
```

2.

```
12345
12345
12345
12345
12345
```

3.

```
1
12
123
1234
12345
```

4.

```
12345
1234
123
12
1
```

5.

```
    5
   45
  345
 2345
12345
```

6.

```
12345
 2345
  345
   45
    5
```

7.

```
    11
   2112
  321123
 43211234
5432112345
5432112345
 43211234
  321123
   2112
    11
```

8.

```
5555555555
5444444445
5433333345
5432222345
5432112345
5432112345
5432222345
5433333345
5444444445
5555555555
```

Bab 6 – Dasar Pemrograman Array dan Array 1 Dimensi

Manfaat Pembelajaran:

1. Mengolah data statistik (nilai total, rata-rata, minimum, maksimul, pencarian dan pengurutan)
2. Simulasi antrian
3. Signal processing

Berikut ini adalah beberapa definisi Array.

1. Sekumpulan data yang memiliki tipe data yang sama tetapi memiliki nilai yang berbeda
2. Dapat berupa array 1 dimensi, 2 dimensi ataupun multidimensi
3. Terdiri dari n buah data untuk array 1 dimensi. Terdiri dari JlhBaris x JlhKolom data untuk array 2 dimensi. Terdiri dari row x col x height data untuk array 3 dimensi dan seterusnya

Pembahasan pada bidang matematika yang mendekati topik array adalah deret dan matrik. Berikut ini adalah contoh pembahasan deret numerik matematika (penggunaan array 1 dimensi) beserta dengan jawabannya.

1. Tuliskan deret bilangan genap sebanyak 10 suku pertama: **2 4 6 8 10 12 14 16 18 20**
2. Tuliskan deret bilangan kelipatan 3 sebanyak 7 suku pertama: **3 6 9 12 15 18 21**
3. Tuliskan deret bilangan kuadrat sebanyak 5 suku pertama: **1 4 9 16 25**
4. Tuliskan deret Fibonacci untuk 10 suku pertama: **1 1 2 3 5 8 11 21 32 53**
5. Tuliskan deret nilai faktorial sebanyak 6 suku pertama: **1 2 6 24 120 720**
6. Tuliskan 10 suku pertama dari persamaan 3x-1 dimana tiap suku genap (bukan suku yang bernilai genap) harus bernilai minus dan tiap suku yang ganjil harus bernilai positif: 2 -5 8 -11 14 -17 20 -23 26 -29
7. Tuliskan urutan proses (langkah-langkah) untuk membalikkan sebuah kata **"belajar"** jika huruf **'b'** dianggap huruf ke 0

huruf **'e'** dianggap huruf ke 1

huruf **'l'** dianggap huruf ke 2

huruf **'a'** dianggap huruf ke 3

huruf **'j'** dianggap huruf ke 4

huruf **'a'** dianggap huruf ke 5

huruf **'r'** dianggap huruf ke 6

Proses pembalikan/pertukaran:

Huruf ke-0 (‘b’) ditukar dengan huruf ke-6 (‘r’) sehingga menjadi “relajab”
Huruf ke-1 (‘e’) ditukar dengan huruf ke-5 (‘a’) sehingga menjadi “ralajbb”
Huruf ke-2 (‘l’) ditukar dengan huruf ke-4 (‘j’) sehingga menjadi “rajaleb”
Huruf ke 3 tidak ditukar karena tidak ada pasangannya atau karena jumlah huruf pada kata **“belajar”** adalah ganjil sehingga hasil pembalikan kata **“belajar”** menjadi **“rajaleb”**

Berikut ini adalah contoh soal pembahasan matrik (array 2 dimensi. NB: tidak disertai jawaban). Tuliskan urutan proses untuk memutar sebuah matrik berukuran 3x4 sebesar 90 derajat ke arah kanan jika matrik asli tersebut memiliki nilai-nilai elemen berikut ini

```
25 68 14 28
74 58 37 19
31 84 70 42
```

Jika diasumsikan bahwa 25 adalah matrik pada elemen [0][0]

Jika diasumsikan bahwa 68 adalah matrik pada elemen [0][1]

Jika diasumsikan bahwa 14 adalah matrik pada elemen [0][2]

Jika diasumsikan bahwa 28 adalah matrik pada elemen [0][3]

Jika diasumsikan bahwa 74 adalah matrik pada elemen [1][0]

Jika diasumsikan bahwa 58 adalah matrik pada elemen [1][1]

Jika diasumsikan bahwa 37 adalah matrik pada elemen [1][2]

Jika diasumsikan bahwa 19 adalah matrik pada elemen [1][3]

Jika diasumsikan bahwa 31 adalah matrik pada elemen [2][0]

Jika diasumsikan bahwa 84 adalah matrik pada elemen [2][1]

Jika diasumsikan bahwa 70 adalah matrik pada elemen [2][2]

Jika diasumsikan bahwa 42 adalah matrik pada elemen [2][3]

Dengan memahami pembahasan soal deret matematika dan matrik pada contoh soal di atas maka makin terlihat bahwa topik-topik array yang akan dibahas berikut ini sangat mendekati soal deret dan matrik pada matematika. Berikut pembahasannya.

Berikut ini adalah karakteristik array berdimensi 1.

1. Terdiri dari n buah data dalam susunan yang berurutan dengan tipe data yang sama.
2. Dapat diakses menggunakan nomor posisi (index) numerik dari 0 s/d n-1.
3. Illustrasi pemetaan array 1 dimensi dengan 10 elemen dapat dilihat pada gambar 6.

0	1	2	3	4	5	6	7	8	9
x	x	x	x	x	x	x	x	x	x

Gambar 6 Ilustrasi pemetaan array 1 dimensi dengan 10 elemen

Berikut ini adalah karakteristik array berdimensi 2.

1. Terdiri dari data sebanyak row x col tersusun secara berurutan dengan tipe data yang sama
2. Dapat diakses dengan menggunakan nomor index dari (0,0); (0,1); s/d (n-1,n-1)
3. Illustrasi pemetaan array 1 dimensi sebanyak 2 x 3 elemen dapat dilihat pada gambar 7

	0	1	2
0	x	x	x
1	x	x	x

Gambar 7 Ilustrasi pemetaan array 2 dimensi dengan 2x3 elemen

Pembagian array berdasarkan alokasi memori.

1. Array Statis

   Contoh:

```
data:Array[0..9] of Integer;   {Bahasa Pascal dan Delphi}
int data[10];                  //Bahasa C/C++
```

2. Array Dinamis

   Contoh:

```
data:Array of Integer          {Bahasa Pascal dan Delphi}
SetLength(data, 10);

int *data=new int[10];       //Bahasa C/C++
int[] data=new int[10];        //Bahasa C/C++, Java dan C#
```

Meskipun array dan list adalah tipe data yang berbeda dari segi kekurangan dan kelebihan, keduanya memiliki kegunaan yang hampir sama sehingga di dalam bahasa pemrograman hanya memiliki tipe data list dan tidak memiiki tipe data array.

Pada dasarnya kegunaan daripada tipe data list hampir sama seperti array hanya saja tipe data array memiliki jumlah elemen statis dimana saat program masih berjalan (runtime) jumlah elemen array statis tidak dapat diubah dan meskipun jumlah elemen array dinamis dapat diubah selama runtime hanya saja isi datanya menjadi hilang, hal demikian tidak terjadi pada tipe data list dinama penambahan dan pengurangan elemen tidak akan mengganggu data yang sudah ada.

Contoh deklarasi dan cara memasukkan data dari keyboard ke dalam array statis berdimensi 1 dalam tipe data list pada bahasa pemrograman Pascal / Delphi.

```
Var data:Array[0..9] of Integer;
    i:Integer;
Begin
 For i := 0 To 9 Do Begin
  Write('Masukkan data ke ', i, ' : ')
  ReadLn(data[i]);
 End;
 WriteLn;
 Write('Data yang berhasil dimasukkan: ');
 For i := 0 To 9 Do
  Console.Write(data[i] & " ")
End.
```

Contoh deklarasi dan cara memasukkan data acak ke dalam array dinamis berdimensi 1 dalam tipe data list pada bahasa pemrograman Pascal / Delphi.

```
Uses SysUtils;  {atau Uses CRT; di Pascal}
Var data:Array[0..9] of Integer;
    i:Integer;
Begin
 Randomize;
 For i := 0 To 9 Do Begin
  data[i]:=Random(100);
  Write(data[i], ' ');
 End;
 WriteLn;
End.
```

Berikut ini adalah permasalahan komputasi yang dapat diselesaikan dengan pemrograman array 1 dimensi beserta dengan dengan pseudocodenya.

1. Total & Average (Rata-Rata)

```
1 total=0
2 FOR i FROM 0 TO n-1
3  data[i]=input()
4  total=total+data[i]
5 END FOR
6 average=total/n
```

Berikut ini adalah simulasi hasil perhitungan nilai total dari pseudo code total dimana kolom C adalah nilai-nilai array yang dimasukkan dari perintah **INPUT()** pada pseudo code baris 3 dan proses perhitungan pada pseudo code baris dapat dilihat pada kolom D dan kolom E.

	A	B	C	D	E	F
1	i	data[i]		rumus total	total	
2					0	0
3	0	data[0]	90	total=total+data[0]	0 + 90 =	90
4	1	data[1]	23	total=total+data[1]	90 + 23 =	113
5	2	data[2]	26	total=total+data[2]	113 + 26 =	139
6	3	data[3]	39	total=total+data[3]	139 + 39 =	178
7	4	data[4]	38	total=total+data[4]	178 + 38 =	216
8	5	data[5]	49	total=total+data[5]	216 + 49 =	265

Dari hasil simulasi perhitungan tersebut dapat dilihat bahwa cara kerja program komputer dalam menghitung total adalah secara bertahap (per setiap data atau satu per satu) dalam waktu yang berbeda atau berdekatan, bukan secara sekaligus dalam waktu yang bersamaan.

2. Minimum & Maximum

```
1 minimum=data[0]
2 FOR i FROM 1 TO n-1
3  IF minimum>data[i] THEN
4   minimum=data[i]
5  END IF
6 END FOR
```

```
1 maximum=data[0]
2 FOR i FROM 1 TO n-1
3  IF maximum<data[i] THEN
4   maximum=data[i]
5  END IF
6 END FOR
```

Berikut ini adalah simulasi proses simulasi pseudo code **nilai minimum** dimana data pada kolom C dianggap telah ada atau telah dimasukkan dari keyboard.

	A	B	C	D	E	F	G
1	i	data[i]		rumus min	min>data[i]	min	keterangan
2							
3	0	data[0]	90	min=data[0]		90	bagian ini tidak berada di dlm looping
4	1	data[1]	23	IF min>data[1] THEN min=data[1]	90 > 23 = True	23	min berubah krn kondisi perbandingan True
5	2	data[2]	26	IF min>data[2] THEN min=data[2]	23 > 26 = False	23	nilai min tdk berubah krn kondisi perbandingan menghasilkan False
6	3	data[3]	39	IF min>data[3] THEN min=data[3]	23 > 39 = False	23	nilai min tdk berubah krn kondisi perbandingan menghasilkan False
7	4	data[4]	38	IF min>data[4] THEN min=data[4]	23 > 38 = False	23	nilai min tdk berubah krn kondisi perbandingan menghasilkan False
8	5	data[5]	49	IF min>data[5] THEN min=data[5]	23 > 49 = False	23	nilai min tdk berubah krn kondisi perbandingan menghasilkan False

Pada simulasi tersebut yaitu pada baris ke 3 dari tabel spreadsheet excel yang merupakan representasi dari baris pertama pseudo code dapat dilihat bahwa rumus min berbeda dari baris 4 s/d baris 8 dari spreadsheet (yang merupakan rpresentasi dari baris ke 2 s/d baris 6 dari pseudo code) karena kode program ini tidak berada di dalam looping for dari baris ke 2 s/d baris 6 dari pseudo code.

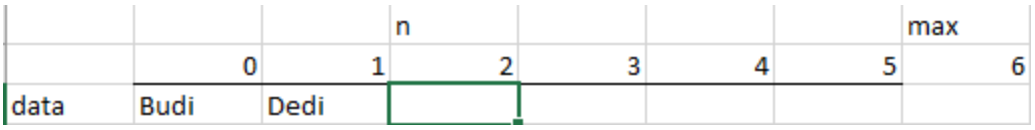
Berikut ini adalah simulasi proses simulasi pseudo code nilai **maksimum** dimana data pada kolom C dianggap telah ada atau telah dimasukkan dari keyboard dimana data[4] diganti dari 49 menjadi 94.

	A	B	C	D	E	F	G
1	i	data[i]		rumus max	max<data[i]	min	keterangan
2							
3	0	data[0]	90	max=data[0]		90	bagian ini tidak berada di dlm looping
4	1	data[1]	23	IF max<data[1] THEN max=data[1]	90 < 23 = False	90	max tdk berubah krn kondisi perbandingan False
5	2	data[2]	26	IF max<data[2] THEN max=data[2]	90 < 26 = False	90	max tdk berubah krn kondisi perbandingan False
6	3	data[3]	39	IF max<data[3] THEN max=data[3]	90 < 39 = False	90	max tdk berubah krn kondisi perbandingan False
7	4	data[4]	38	IF max<data[4] THEN max=data[4]	90 < 38 = False	90	max tdk berubah krn kondisi perbandingan False
8	5	data[5]	94	IF max<data[5] THEN max=data[5]	90 < 94 = True	94	max berubah krn kondisi perbandingan True

Pada simulasi tersebut yaitu pada baris ke 3 dari tabel spreadsheet excel yang merupakan representasi dari baris pertama pseudo code dapat dilihat bahwa rumus max berbeda dari baris 4 s/d baris 8 dari spreadsheet (yang merupakan rpresentasi dari baris ke 2 s/d baris 6 dari pseudo code) karena kode program ini tidak berada di dalam looping for dari baris ke 2 s/d baris 6 dari pseudo code.

3. Insert and Delete Element Simulation

   Untuk mensimulasikan proses insert dan delete akan menggunakan data berikut ini dengan asumsi kapasitas data (max) ada sebanyak 6 dan banyaknya data yang telah masuk ke dalam array adalah n (dengan jumlah data yang telah ada sebanyak 2).



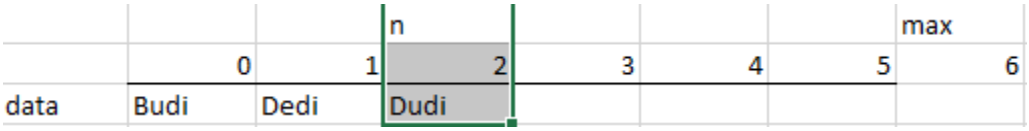
			n				max
	0	1	2	3	4	5	6
data	Budi	Dedi					

a. Sisip belakang

```
1 data[n]=new data
2 n=n+1
```

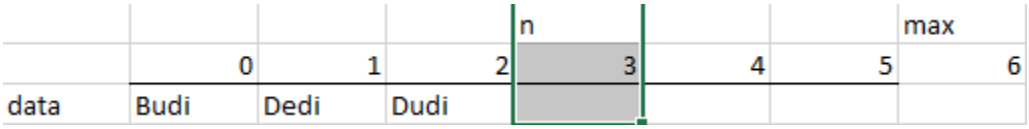
Misalkan data baru yang dimasukkan adalah “Dudi” maka proses simulasinya akan seperti berikut ini.

1 data[n]=databaru (“Dedi”)



			n				max
	0	1	2	3	4	5	6
data	Budi	Dedi	Dudi				

2 n=n+1



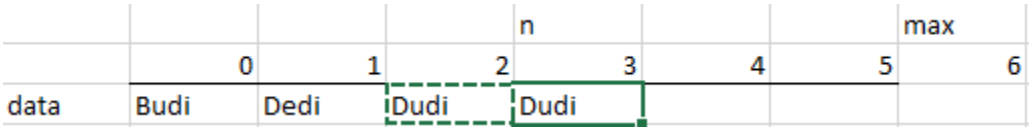
				n			max
	0	1	2	3	4	5	6
data	Budi	Dedi	Dudi				

b. Sisip depan

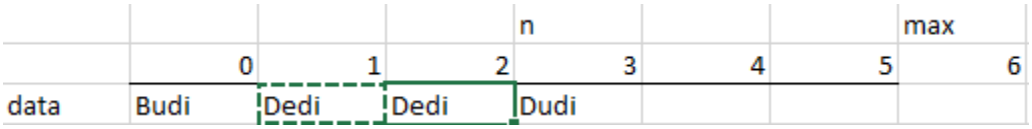
```
1 FOR i FROM n TO 1 STEP -1
2  data[i]=data[i-1]
3 END FOR
4 data[0]=new data
5 n=n+1
```

Misalkan data baru yang dimasukkan adalah “Adi” maka proses simulasinya seperti berikut ini dimana jika banyaknya data atau n adalah sebanyak 3 sehingga banyaknya looping (pseudo code baris 1 s/d baris 3) yang terjadi untuk mengeser data ke posisi sebelahnya (sebelum data ke-0 dimasukkan seperti pseudo code baris 4) juga terjadi sebanyak 3 kali.

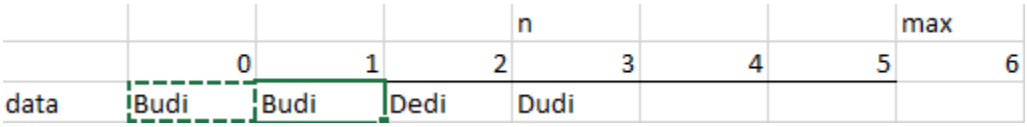
data[i]=data[i-1] -> data[3]=data[2] (baris 2 pseudo code)



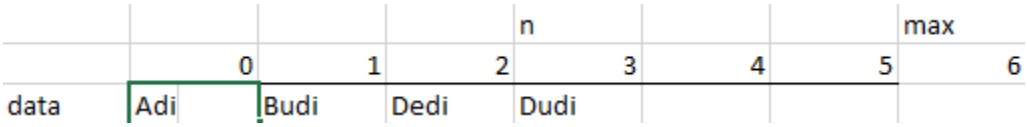
data[i]=data[i-1] -> data[2]=data[1] (baris 2 pseudo code)



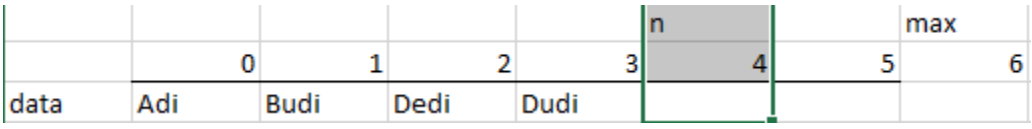
data[i]=data[i-1] -> data[1]=data[0] (baris 2 pseudo code)



data[0]=databaru ("Adi") (baris 4 pseudo code)



n=n+1 (baris 5 pseudo code)



c. Sisip tengah

```
1 FOR i FROM n TO pos+1 STEP -1
2  data[i]=data[i-1]
3 END FOR
4 data[pos]=new data
5 n=n+1
```

Jika data "Cindy" disisipkan pada posisi 2, maka perulangan untuk menggeser data[i]=data[i-1] pada baris ke 2 pseudo code akan terjadi sebanyak 2 kali, karena nilai i

(pada pseudo code baris pertama) akan dimulai dari 4 (n) hingga 2+1 (pos+1) seperti ilustrasi berikut ini.

`data[4]=data[3]`

					n		max
	0	1	2	3	4	5	6
data	Adi	Budi	Dedi	Dudi	Dudi		

`data[3]=data[2]`

					n		max
	0	1	2	3	4	5	6
data	Adi	Budi	Dedi	Dedi	Dudi		

`data[pos]=new data  -> data[2]="Cindy"` (pseudo code baris 4)

					n		max
	0	1	2	3	4	5	6
data	Adi	Budi	CIndy	Dedi	Dudi		

n=n+1 (pseudo code baris 5)

						n	max
	0	1	2	3	4	5	6
data	Adi	Budi	Cindy	Dedi	Dudi		

d. Hapus tengah

```
1 FOR i FROM pos+1 TO n-1
2  data[i-1]=data[i]
3 END FOR
4 n=n-1
```

Misalkan data yang akan dihapus adalah “Cindy” posisi 2 (pos=2)

			pos			n	max
	0	1	2	3	4	5	6
data	Adi	Budi	Cindy	Dedi	Dudi		

Perulangan (looping) yang terjadi pada pseudo code baris ke 2 adalah data[3] diberikan kepada data [2] atau **`data[2]=data[3]`** (looping pertama)

			pos			n	max
	0	1	2	3	4	5	6
data	Adi	Budi	Dedi	Dedi	Dudi		

Selanjutnya masi terjadi pada pseudo code baris 2 adalah data[4] diberikan kepada data [3] atau **`data[3]=data[4]`** (looping kedua)

			pos			n	max
	0	1	2	3	4	5	6
data	Adi	Budi	Dedi	Dudi	Dudi		

Looping hanya terjadi 2 kali karena banyaknya data sebelum pergeseran terjadi adalah 5 dan posisi data yang akan dihapus adalah 2 sehingga sesuai dengan pseudo code baris 1, perulangan dalam mengeser data adalah sebanyak 2 kali yaitu sebanyak n-(pos+1) (5-2+1) yaitu dalam menggeser data[3] kepada data[2] dan data[4] kepada data[3].

Langkah terakhir, nilai n (banyaknya data) dikurangi dengan 1 (pseudo code baris 5)

			pos		n		max
	0	1	2	3	4	5	6
data	Adi	Budi	Dedi	Dudi	Dudi		

Meskipun data[4] (Dudi) sebelumnya masih tertinggal pada posisi 4, namun karena array hanya memperhitungkan data[0] s/d data[n-1], maka data[4] dapat diabaikan saja, karna sifat dari tipe data array adalah statis dimana blok memori yang sudah dipesan untuk array tidak bisa dikembalikan ke sistem operasi begitu saja.

e. Hapus depan

```
1 FOR i FROM 1 TO n-1
2   data[i-1]=data[i]
```

```
3 END FOR
4 n=n-1
```

Jika banyaknya data ada sebanyak 4 dengan kondisi awal data adalah seperti berikut ini dan akan dilakukan proses hapus depan.

					n		max
	0	1	2	3	4	5	6
data	Adi	Budi	Dedi	Dudi			

Maka perulangan untuk menggeser data akan terjadi sebanyak 3 kali karena nilai variable i pada perintah perulangan pseudo code baris 1 akan bergerak dari 1 s/d 3 (1 s/d n-1) sehingga baris 2 pseudo code akan terjadi sebanyak 3 kali sebagai berikut.

`data[0]=data[1]` (pseudo code baris 2)

					n		max
	0	1	2	3	4	5	6
data	Budi	Budi	Dedi	Dudi			

`data[1]=data[2]` (pseudo code baris 2)

					n		max
	0	1	2	3	4	5	6
data	Budi	Dedi	Dedi	Dudi			

`data[2]=data[3]` (pseudo code baris 2)

					n		max
	0	1	2	3	4	5	6
data	Budi	Dedi	Dudi	Dudi			

Dan pada akhirnya jumlah data harus berkurang 1 (pseudo code baris 4)

				n			max
	0	1	2	3	4	5	6
data	Budi	Dedi	Dudi	Dudi			

Seperti yang telah dijelaskan pada proses hapus tengah bahwa data ke 3 (posisi n) dapat diabaikan saja karena data array yang akan diproses adalah data ke 0 s/d n-1.

f. Hapus belakang

```
n=n-1
```

Jika kondisi awal data adalah seperti ilustrasi berikut ini dengan banyaknya data (n) adalah sebanyak 3, maka.

				n			max
	0	1	2	3	4	5	6
data	Budi	Dedi	Dudi				

Maka apabila terjadi hapus belakang, maka n yang bernilai 3 cukup dikurangi dengan 1 menjadi 2 agar data[2] dapat diabaikan seperti ilustrasi berikut ini.

			n	n			max
	0	1	2	3	4	5	6
data	Budi	Dedi	Dudi				

4. Sequential Search

```
1 found=FALSE
2 i=0
3 WHILE i<n AND NOT found
4  IF ValueToSearch=data[i] THEN
5   FoundPosition=I
6   found=TRUE
7  END IF
8  i=i+1
9 END WHILE
```

Ilustrasi kondisi awal

Jika ada sebanyak 6 buah data array dengan kondisi seperti ilustrasi berikut ini dan data akan selalu diperiksa mulai dari posisi 0 (pseudo code baris 2) dimana variabel penanda ditemukan (atau tidak yang dikenal dengan istilah flag pada pseudo code baris 1) dimulai dari FALSE karena pada kondisi awal pencarian belum dilakukan sehingga sudah pasti ditemukan=FALSE berarti data belum ditemukan.

							n
	0	1	2	3	4	5	6
data	90	23	26	39	38	40	

Sebelum memulai ilustrasi proses pencarian dengan sequential search, kita akan mencoba mengganti nama variabel **ValueToSearch** pada pseudo code baris 4 menjadivariabel **cari** dan mengganti nama variabel **found** menjadi **ditemukan**.

Skenario data tidak ditemukan:

Dan jika data yang dicari adalah 50, maka proses pencarian sequential search yang terjadi adalah seperti ilustrasi berikut dan hingga pada akhirnya pada saat i telah mencapai n, hasil perbandingan masih menghasilkan false dan ini artinya data 50 tidak ditemukan.

	A	B	C	D	E
1	i	data[i]		cari=data[i]	ditemukan
2					FALSE
3	0	data[0]	90	hasil perbandingan 50=90 adlh False	masih False
4	1	data[1]	23	hasil perbandingan 50=23 adlh False	masih False
5	2	data[2]	26	hasil perbandingan 50=26 adlh False	masih False
6	3	data[3]	39	hasil perbandingan 50=39 adlh False	masih False
7	4	data[4]	38	hasil perbandingan 50=38 adlh False	masih False
8	5	data[5]	49	hasil perbandingan 50=49 adlh False	masih False

Skenario data ditemukan:

Dan jika data yang dicari adalah 38, maka proses pencarian sequential search yang terjadi adalah seperti ilustrasi berikut dan dapat dilihat bahwa pencarian dimulai dari membandingkan 38 dengan 90 (cari dengan data[0]), kemdian dilanjutkan dengan membandingkan 39 dengan23 (cari dengan data[1]) dan seterusnya dan pada akhirnya data yang dicari yaitu 38 ditemukan pada posisi ke 4.

	A	B	C	D	E	F
1	i	data[i]		cari=data[i]	ditemukan	
2					FALSE	
3	0	data[0]	90	hasil perbandingan 38=90 adlh False	masih False	
4	1	data[1]	23	hasil perbandingan 38=23 adlh False	masih False	
5	2	data[2]	26	hasil perbandingan 38=26 adlh False	masih False	
6	3	data[3]	39	hasil perbandingan 38=39 adlh False	masih False	
7	4	data[4]	38	hasil perbandingan 38=38 adlh True	menjadi True	
8	5	data[5]	49	hentikan perulangan dan tidak perlu		
9			diperiksa lagi karena data sudah ketemu			

Ketika data ditemukan pada posisi ke 4, maka perulangan akan dihentikan, ini disebabkan oleh kondisi perulangan pada pseudo code baris 3 tertulis bahwa kondisi perulangan adalah selama I masi lebih kecil dari banyaknya data n (i<n) dan selama belum ditemukan (not found).

5. Bubble Sort

```
1 FOR r = 0 TO n-2
2  FOR i = r+1 TO n-1
3   IF data[r]>data[i] THEN
4    Swap(data[r], data[i])
5   END IF
6  END FOR
7 END FOR
```

Berikut ini adalah ilustrasi dari proses pengurutan pseudo code bubble sort

Pada saat **r=0** (round 0) i dimulai dari **1 s/d 5** -> **r+1 s/d n-1**

	r	i					n
	0	1	2	3	4	5	6
data	90	23	26	39	38	49	

Perbandingan yang terjadi adalah apakah data[r] > data[i] (pseudo code baris 3) sehingga yang dibandingkan adalah 90 > 23 (data[0] > data[1]). Karena 90 > 23 menghasilkan true, maka proses penukaran data pada pseudo code baris 4 akan terjadi sehingga letak 90 dan 23 akan tertukar menjadi

	r	i					n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Masih di r=0 tetapi kali ini i bergerak dari nilai 1 menjadi 2.

	r		i				n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Perbandingan tetap terjadi adalah data[r] > data[i] tetapi dengan nilai i yang sudah bernilai 2, sehingga perbandingan yang terjadi adalah 23 > 26 (data[0] > data[2]). Karena 23 > 26 menghasilkan false, maka proses penukaran data pada pseudo code baris 4 tidak terjadi sehingga letak 23 dan 26 tidak terjadi.

Masih di r=0 tetapi kali ini i bergerak dari nilai 1 menjadi 3.

	r			i			n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Selanjutnya perbandingan terjadi adalah 23 > 39 (pseudo code baris 3) yang mengashilkan false sehingga pertukaran tidak terjadi.

Pertukaran juga tidak terjadi pada saat i=4 dan i=5 pada round 0 karena data[r] > data[i] menghasilkan false seperti 2 buah ilustrasi berikut ini.

	r			i			n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

	r			i			n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Berpindah ke round 1 **(r=1),** i dimulai dari **2 s/d 5 -> r+1 s/d n-1**

		r	i				n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Perbandingan yang terjadi adalah 90 > 26 (pseudo code baris 3) dan menghasilkan true sehingga letak 90 dan 26 akan bertukar (pseudo code baris 4) menjadi

		r	i				n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Masih di r=1 tetapi kali ini i bergerak dari nilai 2 menjadi 3.

		r		i			n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Pertukaran tidak terjadi karena 26 > 39 menghasilkan false, begitu juga dengan perbandingan 26 > 38 dan 26 > 49 juga menghasilkan false seperti ilustrasi berikut ini sehingga tidak terjadi pertukaran.

		r			i		n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

		r				i	n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Berpindah ke round 1 **(r=2),** i dimulai dari **3 s/d 5 -> r+1 s/d n-1**

			r	i			n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Perbandingan yang terjadi adalah 90 > 39 (pseudo code baris 3) dan menghasilkan true sehingga letak 90 dan 39 akan bertukar (pseudo code baris 4) menjadi

			r	i			n
	0	1	2	3	4	5	6
data	23	26	39	90	38	49	

Masih di r=2 tetapi kali ini i bergerak dari nilai 3 menjadi 4.

			r		i		n
	0	1	2	3	4	5	6
data	23	26	39	90	38	49	

Selanjutnya perbandingan yang terjadi adalah 39 > 38 (pseudo code baris 3) dan menghasilkan true sehingga letak 39 dan 38 harus ditukar menjadi

			r		i		n
	0	1	2	3	4	5	6
data	23	26	38	90	39	49	

Saat r=2 dan i=5. Pertukaran tidak terjadi karena 38 < 49 menghasilkan false.

			r			i	n
	0	1	2	3	4	5	6
data	23	26	38	90	39	49	

Berpindah ke round 1 **(r=3),** i dimulai dari **4 s/d 5 -> r+1 s/d n-1**

				r	i		n
	0	1	2	3	4	5	6
data	23	26	38	90	39	49	

Perbandingan yang terjadi adalah 90 > 39 (pseudo code baris 3) dan menghasilkan true sehingga letak 90 dan 39 akan bertukar (pseudo code baris 4) menjadi

				r	i		n
	0	1	2	3	4	5	6
data	23	26	38	39	90	49	

NB: Kelihatan pada beberapa ilustrasi yang ada bahwa 90 telah bertukar dengan 39 berkali-kali dan ini yang menjadi kekurangan dari algoritma bubble sort.

Berpindah ke round 1 **(r=4),** i dimulai dari 5 **s/d 5 -> r+1 s/d n-1**

					r	i	n
	0	1	2	3	4	5	6
data	23	26	38	39	90	49	

Perbandingan yang terjadi adalah 90 > 49 (pseudo code baris 3) dan menghasilkan true sehingga letak 90 dan 49 akan bertukar (pseudo code baris 4) menjadi

					r	i	n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

6. Insertion Sort

```
1 FOR r = n-2 TO 0 STEP -1
2  FOR i = 0 TO r
3   j=i+1
4   IF data[i]>data[j] THEN
5    Swap(data[i], data[j])
6   END IF
7  END FOR
8 END FOR
```

Berikut ini adalah kondisi awal dari ilustrasi proses pengurutan pseudo code insertion sort.

							n
	0	1	2	3	4	5	6
data	90	23	26	39	38	49	

Pada baris pertama pseudo code, nilai r akan dimulai dari n-2 sampai dengan 0 sehingga jika banyaknya ada ada sebanyak 6, maka perulangan r akan diulang dari r=6-4 s/d 0 sehingga jumlah round ada sebanyak 5 kali.

Pada round 4, perulangan i pada pseudo code baris 2 akan dimulai dari 0 s/d 4 (0 s/d r).

Pada saat i=0, yang perbandingan yang terjadi 90 > 23 (data[0] > data[1] pada pseudo code baris 4)

	i	j=i+1			r		n
	0	1	2	3	4	5	6
data	90	23	26	39	38	49	

Karena 90 > 23 menghasilkan true maka kedua data tersebut harus ditukar menjadi

	i	j=i+1			r		n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Selanjutnya i bergerak dari 0 menjadi 1 sehingga j akan bernilai i+1 = 2 (pseudo code baris 3) sehingga perbandingan yang terjadi adalah 90 > 26 (data[1] > data[2]).

		i	j=i+1		r		n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Karena 90 > 26 menghasilkan true maka kedua data tersebut harus ditukar menjadi

		i	j=i+1		r		n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Selanjutnya i bergerak dari 1 menjadi 2 sehingga j akan bernilai i+1 = 3 (pseudo code baris 3) sehingga perbandingan yang terjadi adalah 90 > 39 (data[2] > data[3]).

			i	j=i+1	r		n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Karena 90 > 39 menghasilkan true maka kedua data tersebut harus ditukar menjadi

			i	j=i+1	r		n
	0	1	2	3	4	5	6
data	23	26	39	90	38	49	

Selanjutnya i bergerak dari 2 menjadi 3 sehingga j akan bernilai i+1 = 4 (pseudo code baris 3) sehingga perbandingan yang terjadi adalah 90 > 38 (data[3] > data[4]).

				i	j=i+1 == r		n
	0	1	2	3	4	5	6
data	23	26	39	90	38	49	

Karena 90 > 38 menghasilkan true maka kedua data tersebut harus ditukar menjadi

				i	j=i+1 == r		n
	0	1	2	3	4	5	6
data	23	26	39	38	90	49	

Selanjutnya i bergerak dari 3 menjadi 4 sehingga j akan bernilai i+1 = 5 (pseudo code baris 3) sehingga perbandingan yang terjadi adalah 90 > 49 (data[4] > data[5]).

					i == r	j=i+1	n
	0	1	2	3	4	5	6
data	23	26	39	38	90	49	

Karena 90 > 49 menghasilkan true maka kedua data tersebut harus ditukar menjadi

					i == r	j=i+1	n
	0	1	2	3	4	5	6
data	23	26	39	38	49	90	

Proses i terakhir pada saat r=4, nilai i sudah sama dengan r yaitu 4 (pseudo code baris 2), sehingga perulangan i harus berhenti dan perulangan r akan dilanjutkan dari r=4 menjadi r=3 dan karena perulangan r mengulangi perulangan I (pseudo code baris 2) maka perulangan i dikerjakan kembali dimana I kembali berulang dari 0.

	i	j=i+1		r			n
	0	1	2	3	4	5	6
data	23	26	39	38	49	90	

Karena 23 > 26 menghasilkan false maka kedua data tersebut tidak terjadi pertukaran

Selanjutnya i bergerak dari 0 menjadi 1 sehingga j akan bernilai i+1 = 2 (pseudo code baris 3) sehingga perbandingan yang terjadi adalah 26 > 39 (data[1] > data[2]).

		i	j=i+1	r			n
	0	1	2	3	4	5	6
data	23	26	39	38	49	90	

Karena 26 > 39 menghasilkan false maka kedua data tersebut tidak terjadi pertukaran

Selanjutnya i bergerak dari 1 menjadi 2 sehingga j akan bernilai i+1 = 3 (pseudo code baris 3) sehingga perbandingan yang terjadi adalah 39 > 38 (data[2] > data[3]).

			i	j=i+1 == r			n
	0	1	2	3	4	5	6
data	23	26	39	38	49	90	

Karena 39 > 38 menghasilkan true maka kedua data tersebut harus ditukar menjadi

			i	j=i+1 == r			n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Selanjutnya i bergerak dari 2 menjadi 3 sehingga j akan bernilai i+1 = 4 (pseudo code baris 3) sehingga perbandingan yang terjadi adalah 39 > 49 (data[3] > data[4]).

				i == r	j=i+1		n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Karena 39 > 49 menghasilkan false maka kedua data tersebut tidak terjadi pertukaran

Proses i terakhir pada saat r=3, nilai i sudah sama dengan r yaitu 3 (pseudo code baris 2), sehingga perulangan i harus berhenti dan perulangan r akan dilanjutkan dari r=3 menjadi r=2 dan karena perulangan r mengulangi perulangan I (pseudo code baris 2) maka perulangan i dikerjakan kembali dimana i kembali berulang dari 0.

	i	j=i+1	r				n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Karena 23 > 26 menghasilkan false maka kedua data tersebut tidak terjadi pertukaran

Selanjutnya i bergerak dari 0 menjadi 1 sehingga j akan bernilai i+1 = 2 (pseudo code baris 3) sehingga perbandingan yang terjadi adalah 26 > 38 (data[1] > data[2]).

		i	j=i+1 == r				n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Karena 26 > 38 menghasilkan false maka kedua data tersebut tidak terjadi pertukaran

Selanjutnya i bergerak dari 1 menjadi 2 sehingga j akan bernilai i+1 = 3 (pseudo code baris 3) sehingga perbandingan yang terjadi adalah 38 > 39 (data[2] > data[3]).

			i == r	j=i+1			n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Karena 38 > 39 menghasilkan false maka kedua data tersebut tidak terjadi pertukaran

Proses i terakhir pada saat r=2, nilai i sudah sama dengan r yaitu 2 (pseudo code baris 2), sehingga perulangan i harus berhenti dan perulangan r akan dilanjutkan dari r=2 menjadi r=1 dan karena perulangan r mengulangi perulangan I (pseudo code baris 2) maka perulangan i dikerjakan kembali dimana i kembali berulang dari 0.

	i	j=i+1 == r					n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Karena 23 > 26 menghasilkan false maka kedua data tersebut tidak terjadi pertukaran

Selanjutnya i bergerak dari 0 menjadi 1 sehingga j akan bernilai i+1 = 2 (pseudo code baris 3) sehingga perbandingan yang terjadi adalah 26 > 38 (data[1] > data[2]).

		i == r	j=i+1				n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Karena 26 > 38 menghasilkan false maka kedua data tersebut tidak terjadi pertukaran

Proses i terakhir pada saat r=1, nilai i sudah sama dengan r yaitu 1 (pseudo code baris 2), sehingga perulangan i harus berhenti dan perulangan r akan dilanjutkan dari r=1 menjadi r=0 dan karena perulangan r mengulangi perulangan I (pseudo code baris 2) maka perulangan i dikerjakan kembali dimana i kembali berulang dari 0.

	I == r	j=i+1					n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Karena 23 > 26 menghasilkan false maka kedua data tersebut tidak terjadi pertukaran

Demikian ilustrasi proses pengurutan dengan menggunakan algoritma insertion. Kekurangan dari metode ini adalah, perbandingan antara 23 dan 26 terjadi berulang-ulang padahal letak kedua data tersebut sudah sesuai.

7. Selection Sort

```
1 FOR r = 0 TO n-2
```

```
2   k=r
3   FOR i = r+1 TO n-1
4    IF data[k]>data[i] THEN
5     k=i
6    END IF
7   END FOR
8   Swap(data[r], data[k])
9 END FOR
```

Berikut ini adalah ilustrasi dari proses pengurutan pseudo code selection sort

Pada saat **r=0** (round 0) i dimulai dari **1 s/d 5 -> r+1 s/d n-1**

Sebelum perulangan i dimulai, variabel k akan mengikuti nilai r (pseudo code baris 2). Tugas dari variabel k adalah akan selalu menuju ke posisi data terkecil yang didapatkan sehingga di sini nilai k adalah 0.

	k = r	i					n
	0	1	2	3	4	5	6
data	90	23	26	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 90 > 23 (data[0] > data[1]). Karena 90 > 23 menghasilkan **true**, nilai k akan dipindahkan ke posisi I yaitu 1 (pseudo code baris 5).

	r	k = i					n
	0	1	2	3	4	5	6
data	90	23	26	39	38	49	

Masih di r=0 tetapi kali ini i bergerak dari nilai 1 menjadi 2.

	r	k	i				n
	0	1	2	3	4	5	6
data	90	23	26	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 23 > 26 (data[1] > data[2]). Karena 23 > 26 menghasilkan **false** maka nilai k tidak berubah dan tetap bernilai 1.

Masih di r=0 tetapi kali ini i bergerak dari nilai 2 menjadi 3.

	r	k		i			n
	0	1	2	3	4	5	6
data	90	23	26	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 23 > 39 (data[1] > data[3]). Karena 23 > 39 menghasilkan **false** maka nilai k tidak berubah dan tetap bernilai 1.

Masih di r=0 tetapi kali ini i bergerak dari nilai 3 menjadi 4.

	r	k			i		n
	0	1	2	3	4	5	6
data	90	23	26	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 23 > 38 (data[1] > data[4]). Karena 23 > 38 menghasilkan **false** maka nilai k tidak berubah dan tetap bernilai 1.

Masih di r=0 tetapi kali ini i bergerak dari nilai 4 menjadi 5.

	r	k				i	n
	0	1	2	3	4	5	6
data	90	23	26	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 23 > 49 (data[1] > data[5]). Karena 23 > 49 menghasilkan **false** maka nilai k tidak berubah dan tetap bernilai 1.

Setelah perulangan I selesai, maka data[r] akan ditukar dengan data[k] (pseudo code baris 8)

	r	k					n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Pada saat **r=1** (round 1) i dimulai dari **2 s/d 5 -> r+1 s/d n-1**

Sebelum perulangan i dimulai, variabel k akan mengikuti nilai r (pseudo code baris 2) yaitu 1.

		k = r	i				n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 90 > 26 (data[1] > data[2]). Karena 90 > 26 menghasilkan **true**, nilai k akan dipindahkan ke posisi i yaitu 2 (pseudo code baris 5).

		r	k = i				n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Masih di r=1 tetapi kali ini i bergerak dari nilai 2 menjadi 3.

		r	k	i			n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 26 > 39 (data[2] > data[3]). Karena 26 > 39 menghasilkan **false**, maka nilai k tidak berubah dan tetap bernilai 2.

Masih di r=1 tetapi kali ini i bergerak dari nilai 3 menjadi 4.

		r	k		i		n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 26 > 38 (data[2] > data[4]). Karena 26 > 38 menghasilkan **false**, maka nilai k tidak berubah dan tetap bernilai 2.

Masih di r=1 tetapi kali ini i bergerak dari nilai 4 menjadi 5.

		r	k			i	n
	0	1	2	3	4	5	6
data	23	90	26	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 26 > 49 (data[2] > data[5]). Karena 26 > 49 menghasilkan **false**, maka nilai k tidak berubah dan tetap bernilai 2.

Setelah perulangan i selesai, maka data[r] akan ditukar dengan data[k] (pseudo code baris 8)

		r	k			i	n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Pada saat **r=2** (round 2) i dimulai dari **3 s/d 5 -> r+1 s/d n-1**

Sebelum perulangan i dimulai, variabel k akan mengikuti nilai r (pseudo code baris 2) yaitu 2.

			k = r	i			n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 90 > 39 (data[2] > data[3]). Karena 90 > 39 menghasilkan **true**, nilai k akan dipindahkan ke posisi i yaitu 3 (pseudo code baris 5).

			r	k = i			n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Masih di r=2 tetapi kali ini i bergerak dari nilai 3 menjadi 4.

			r	k	i		n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 39 > 38 (data[3] > data[4]). Karena 39 > 38 menghasilkan **true** maka nilai k akan mengikuti nilai i menjadi 4 (pseudo code baris 5).

Masih di r=2 tetapi kali ini i bergerak dari nilai 4 menjadi 5.

			r		k	i	n
	0	1	2	3	4	5	6
data	23	26	90	39	38	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 38 > 49 (data[4] > data[5]). Karena 38 > 49 menghasilkan **false** sehingga nilai k tidak berubah dan tetap bernilai 4.

Setelah perulangan i selesai, maka data[r] akan ditukar dengan data[k] (pseudo code baris 8)

			r		k	i	n
	0	1	2	3	4	5	6
data	23	26	39	39	90	49	

Pada saat **r=3** (round 3) i dimulai dari **4 s/d 5 -> r+1 s/d n-1**

Sebelum perulangan i dimulai, variabel k akan mengikuti nilai r (pseudo code baris 2) yaitu 3.

				k = r	i		n
	0	1	2	3	4	5	6
data	23	26	38	39	90	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 39 > 90 (data[3] > data[4]). Karena 39 > 90 menghasilkan **false** sehingga nilai k tidak berubah dan tetap bernilai 3.

Masih di r=3 tetapi kali ini i bergerak dari nilai 4 menjadi 5.

				k = r		i	n
	0	1	2	3	4	5	6
data	23	26	38	39	90	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 39 > 49 (data[3] > data[5]). Karena 39 > 49 menghasilkan **false** sehingga nilai k tidak berubah dan tetap bernilai 3.

Setelah perulangan i selesai, maka data[r] akan ditukar dengan data[k] (pseudo code baris 8)

Tetapi karena posisi k dan r adalah sama, maka nilai 39 tertukar pada posisi yang sama sehingga data hasil pertukaran masih sama persis dengan kondisi terakhir.

Pada saat **r=4** (round 4) i dimulai dari **5 s/d 5 -> r+1 s/d n-1**

Sebelum perulangan i dimulai, variabel k akan mengikuti nilai r (pseudo code baris 2) yaitu 4.

					k = r	i	n
	0	1	2	3	4	5	6
data	23	26	38	39	90	49	

Perbandingan yang terjadi adalah apakah data[k] > data[i] (pseudo code baris 4) sehingga yang dibandingkan adalah 90 > 49 (data[4] > data[5]). Karena 90 > 49 menghasilkan **true** maka nilai k akan mengikuti nilai i menjadi 5 (pseudo code baris 5).

					r	k	n
	0	1	2	3	4	5	6
data	23	26	38	39	90	49	

Setelah perulangan i selesai, maka data[r] akan ditukar dengan data[k] (pseudo code baris 8)

					r	k	n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Demikian proses pengurutan algoritma selection dimana cara kerja selection sort adalah memilih yang terkecil pertama untuk digantikan dengan data di posisi pertama, memilih yang terkecil kedua untuk digantikan dengan data di posisi kedua dan seterusnya. Kelebihan dari algoritma ini dibandingkan dengan 2 algoritma pengurutan sebelumnya adalah: pertukaran hanya terjadi seperlunya saja.

8. Shell Sort

```
1 FOR j = INT(n/2) TO 1 STEP -1
2  FOR i = 0 TO n-j-1
3   k=i+j
4   IF data[i]>data[k] THEN
5    Swap(data[i], data[k])
```

```
6    END IF
7   END FOR
8 END FOR
```

Jika banyaknya data (n) ada sebanyak 6 maka perulangan j (pseudo code baris 1) diulang dari 3 s/d 1.

Pada saat j bernilai 3, perulangan i dimulai dari 0 s/d 2 (pseudo code baris 1)

Pada saat nilai i=0 nilai k adalah 3 karena k=i+j = 0+3 = 3 (pseudo code baris 3).

	i			k=i+j = j			n
	0	1	2	3	4	5	6
data	90	23	26	39	38	49	

Perbandingan yang terjadi adalah 90 > 39 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **true**. Sehingga letak 90 dan 39 akan ditukar pada pseudo code baris 5 menjadi

	i			k=i+j = j			n
	0	1	2	3	4	5	6
data	39	23	26	90	38	49	

Pada saat nilai i=1 nilai k adalah 4 karena k=i+j = 1+3 = 4 (pseudo code baris 3).

		i		j	k=i+j		n
	0	1	2	3	4	5	6
data	39	23	26	90	38	49	

Perbandingan yang terjadi adalah 23 > 38 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **false**. Sehingga data 23 dan 38 tidak tertukar.

Pada saat nilai i=2 nilai k adalah 4 karena k=i+j = 2+3 = 5 (pseudo code baris 3).

			i	j		k=i+j	n
	0	1	2	3	4	5	6
data	39	23	26	90	38	49	

Perbandingan yang terjadi adalah 26 > 49 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **false**. Sehingga data 26 dan 49 tidak tertukar.

Pada saat j bernilai 2, perulangan i dimulai dari 0 s/d 3 (pseudo code baris 1)

Pada saat nilai i=0 nilai k adalah 3 karena k=i+j = 0+2 = 2 (pseudo code baris 3).

	i		k=i+j =j			k=i+j	n
	0	1	2	3	4	5	6
data	39	23	26	90	38	49	

Perbandingan yang terjadi adalah 39 > 26 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **true**. Sehingga data 39 dan 26 menjadi tertukar.

	i		k=i+j =j			k=i+j	n
	0	1	2	3	4	5	6
data	26	23	39	90	38	49	

Pada saat nilai i=1 nilai k adalah 3 karena k=i+j = 1+2 = 3 (pseudo code baris 3).

		i	j	k=i+j			n
	0	1	2	3	4	5	6
data	26	23	39	90	38	49	

Perbandingan yang terjadi adalah 23 > 90 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **false**. Sehingga data 23 dan 90 tidak tertukar.

Pada saat nilai i=2 nilai k adalah 4 karena k=i+j = 2+2 = 4 (pseudo code baris 3).

			i == j		k=i+j		n
	0	1	2	3	4	5	6
data	26	23	39	90	38	49	

Perbandingan yang terjadi adalah 39 > 38 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **true**. Sehingga data 39 dan 38 menjadi tertukar.

			i == j		k=i+j		n
	0	1	2	3	4	5	6
data	26	23	38	90	39	49	

Pada saat nilai i=3 nilai k adalah 5 karena k=i+j = 3+2 = 5 (pseudo code baris 3).

			j	i		k=i+j	n
	0	1	2	3	4	5	6
data	26	23	38	90	39	49	

Perbandingan yang terjadi adalah 90 > 49 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **true**. Sehingga data 90 dan 49 menjadi tertukar.

			j	i		k=i+j	n
	0	1	2	3	4	5	6
data	26	23	38	49	39	90	

Pada saat j bernilai 1, perulangan i dimulai dari 0 s/d 4 (pseudo code baris 1)

Pada saat nilai i=0 nilai k adalah 1 karena k=i+j = 0+1 = 1 (pseudo code baris 3).

	i	k=i+j == j					n
	0	1	2	3	4	5	6
data	26	23	38	49	39	90	

Perbandingan yang terjadi adalah 26 > 23 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **true**. Sehingga data 26 dan 23 menjadi tertukar.

	i	k=i+j == j					n
	0	1	2	3	4	5	6
data	23	26	38	49	39	90	

Pada saat nilai i=1 nilai k adalah 2 karena k=i+j = 1+1 = 2 (pseudo code baris 3).

		i == j	k=i+j				n
	0	1	2	3	4	5	6
data	23	26	38	49	39	90	

Perbandingan yang terjadi adalah 26 > 38 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **false**. Sehingga data 26 dan 38 tidak tertukar.

Pada saat nilai i=2 nilai k adalah 3 karena k=i+j = 1+1 = 3 (pseudo code baris 3).

		j	i	k=i+j			n
	0	1	2	3	4	5	6
data	23	26	38	49	39	90	

Perbandingan yang terjadi adalah 38 > 49 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **false**. Sehingga data 38 dan 49 tidak tertukar.

Pada saat nilai i=3 nilai k adalah 4 karena k=i+j = 1+1 = 4 (pseudo code baris 3).

		j		i	k=i+j		n
	0	1	2	3	4	5	6
data	23	26	38	49	39	90	

Perbandingan yang terjadi adalah 49 > 39 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **true**. Sehingga data 49 dan 39 menjadi tertukar.

Pada saat nilai i=4 nilai k adalah 5 karena k=i+j = 1+1 = 5 (pseudo code baris 3).

		j			i	k=i+j	n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Perbandingan yang terjadi adalah 49 > 90 (data[i] > data[k]) dan hasil perbandingannya menghasilkan **true**. Sehingga data 49 dan 90 menjadi tertukar.

9. Median atau nilai tengah (NB: data harus dalam keadaan sudah diurutkan)

```
1 midpos=INT(n/2)
2 IF n MOD 2 = 0 THEN
3  median=(data[midpos-1]+data[midpos])/2
4 ELSE
5  median=data[midpos]
6 END IF
```

Dari pseudo code di atas dapat dilihat bahwa perhitungan median menjadi berbeda ketika banyaknya ada adalah ganjil dengan ketika banyaknya data adalah genap. Sebagai contoh jika jumlah data ada sebanyak 6 maka nilai tengahnya diperoleh dengan menjumlahkan data[2] dan data[3] dan dibagi dengan 2. Karena variable midpos akan memperoleh nilai 3 yang didapatkan dari banyaknya data dibagi dengan 2 (INT(6/2)) maka data yang diambil adalah data[2] dan data [3] sehingga pada simulasi di di bawah ini nilai tengah yang didapatkan adalah (38+39)/2=38.5.

				mid=int(n/2)			n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Sementara itu jika banyaknya adalah sebanyak 5, maka variable midpos akan bernilai 2 (INT(5/2)) karena nilai pembulatan dari 5/2 adalah 2 sehingga nilai tengah yang diambil adalah nilai ke 2. Karena jumlah data adalah ganjil, maka nilai tengah hanya ada 1

sedangkan jika jumlah data adalah genap maka ada 2 nilai tengah yang harus di-rata-rata-kan.

			mid=int(n/2)			n
	0	1	2	3	4	5
data	23	26	38	39	49	

10. Binary Search

```
 1 left=0
 2 right=n-1
 3 found=FALSE
 4 WHILE left<=right AND NOT found
 5  middle=(left+right)/2
 6  IF ValueToSearch=data[middle] THEN
 7   found=TRUE
 8  ELSE IF ValueToSearch<data[middle] THEN
 9   right=middle-1
10  ELSE
11   left=middle+1
12  END IF
13 END WHILE
14 IF found THEN FoundPosition=middle ELSE FoundPosition=-1
```

Agar dapat menerapkan algoritma binary search maka ada 1 syarat yang harus dipenuhi yaitu data sudah harus dalam keadaan terurut. Berikut ini adalah kondisi awal data yang sudah terurut beserta dengan kondisi variabel left (pseudo code baris 1), right (pseudo code baris 2) dan middle (pseudo code baris 5),

	left		mid			right	n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Kita mulai dari kondisi data ditemukan dalam kasus yang terburuk yaitu ketika data yang dicari adlh 90 (ValueToSearch) yaitu data yang ditemukan di posisi terakhir. Jika

menggunakan algoritma sequential search, maka proses perbandingan untuk menemukan data[5] adalah 6 kali.

Pada pseudo code baris 6, yang terjadi adlh apakah 90 == 38 (ValueToSearch==data[middle]) karena middle bernilai 2 maka yang akan dibandingkan dengan 90 adalah 38 dan hasil perbandingan menghasilkan **false,** sehingga pseudo code baris 7 tidak dijalankan dan pseudo code baris 8 akan dijalankan. Pada pseudo code baris 8, perbandingan yang terjadi adalah apakah 90<38 dan hasil perbandingan juga menghasikan **false** sehingga pseudo code baris 9 tidak dijalankan tetapi pseudo code baris 10 dan 11 yang akan dijalankan sehingga **left** akan bernilai 2+1 = 3 karena digeser menjadi **mid-1** sesuai dengan pseudo code baris 11.

Perulangan berlanjut pada pseudo code baris 4 dimana left<=right masih menghasilkan nilai **true** karena pada looping sebelumnya left bernilai 3 dan right masih bernilai 5 sehingga perbandingan 3<=5 akan menghasilkan true. Karena kondisi perbandingan while menghasilkan **true** maka pseudo code baris 5 akan dijalankan yaitu mid=(left+right)/2 = (3+5)/2 = 4.

				left	mid	right	n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Pada pseudo code baris 6, yang terjadi adlh apakah 90 == 49 (ValueToSearch==data[middle]) karena middle bernilai 4 maka yang akan dibandingkan dengan 90 adalah 49 dan hasil perbandingan menghasilkan **false,** sehingga pseudo code baris 7 tidak dijalankan dan pseudo code baris 8 akan dijalankan. Pada pseudo code baris 8, perbandingan yang terjadi adalah apakah 90<49 dan hasil perbandingan juga menghasikan **false** sehingga pseudo code baris 9 tidak dijalankan tetapi pseudo code baris 10 dan 11 yang akan dijalankan sehingga **left** akan bernilai 4+1 = 5 karena digeser menjadi **mid-1** sesuai dengan pseudo code baris 11.

Perulangan berlanjut pada pseudo code baris 4 dimana left<=right masih menghasilkan nilai **true** karena pada looping sebelumnya left bernilai 5 dan right masih bernilai 5 sehingga

perbandingan 5<=5 akan menghasilkan true. Karena kondisi perbandingan while menghasilkan **true** maka pseudo code baris 5 akan dijalankan yaitu mid=(left+right)/2 = (5+5)/2 = 5.

						lef=t=right==mid	
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Pada pseudo code baris 6, yang terjadi adlh apakah 90 == 90 (ValueToSearch==data[middle]) karena middle bernilai 5 maka yang akan dibandingkan dengan 90 adalah 90 dan hasil perbandingan menghasilkan **true,** sehingga pseudo code baris 7 akan dijalankan sehingga variabel found akan bernilai **true**. Dengan dijalankan pseudo code baris 7 maka pseudo code baris 8 s/d 12 akan diabaikan sehingga looping akan berulang ke pseudocode baris 4 yaitu pada perintah **`WHILE left<=right AND NOT found`**. Dengan variabel found yang sudah bernilai **true** menyebabkan kondisi perulangan while tersebut menjadi false sehingga perulangan while akan dihentikan dan posisi data ditemukan sudah tercatat pada variabel middle, yaitu ditemukan pada posisi 5.

Selanjutnya kita bahas skenario data tidak ditemukan, yaitu ketika data yang di cari (ValueToSearch) adalah 25. Dimana kondisi awal setelah pseudo code baris 4 dan 5 dijalankan adalah seperti berikut ini.

	left		mid			right	n
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Pada pseudo code baris 6, yang terjadi adlh apakah 25 == 38 (ValueToSearch==data[middle]) karena middle bernilai 2 maka yang akan dibandingkan dengan 25 adalah 38 dan hasil perbandingan menghasilkan **false,** sehingga pseudo code baris 7 tidak dijalankan dan pseudo code baris 8 akan dijalankan. Pada pseudo code baris 8, perbandingan yang terjadi adalah apakah 25<38 dan hasil perbandingan menghasikan **true** sehingga pseudo code baris 9 akan dijalankan dan pseudo code baris 10 dan 11 tidak akan dijalankan sehingga **right** akan bernilai 2-1 = 1 karena digeser menjadi **mid-1** sesuai dengan pseudo code baris 9.

Perulangan berlanjut pada pseudo code baris 4 dimana left<=right masih menghasilkan nilai **true** karena pada looping sebelumnya left bernilai masih bernilai 0 dan right sudah berubah menjadi 0 sehingga perbandingan 0<=1 akan menghasilkan true. Karena kondisi perbandingan while menghasilkan **true** maka pseudo code baris 5 akan dijalankan yaitu mid=(left+right)/2 = (0+1)/2 = 0.5 dan dibulatkan menjadi 0.

	left==mid	right					
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Pada pseudo code baris 6, yang terjadi adlh apakah 25 == 23 (ValueToSearch==data[middle]) karena middle bernilai 2 maka yang akan dibandingkan dengan 25 adalah 23 dan hasil perbandingan menghasilkan **false,** sehingga pseudo code baris 7 tidak dijalankan dan pseudo code baris 8 akan dijalankan. Pada pseudo code baris 8, perbandingan yang terjadi adalah apakah 25<23 dan hasil perbandingan menghasikan **false** sehingga pseudo code baris 9 tidak akan dijalankan tetapi pseudo code baris 10 dan 11 yang akan dijalankan sehingga **left** akan bernilai 0+1 = 1 karena digeser menjadi **mid+1** sesuai dengan pseudo code baris 9.

Perulangan berlanjut pada pseudo code baris 4 dimana left<=right masih menghasilkan nilai **true** karena pada looping sebelumnya left bernilai masih bernilai 1 dan right sudah berubah menjadi 1 sehingga perbandingan 1<=1 akan menghasilkan true. Karena kondisi perbandingan while menghasilkan **true** maka pseudo code baris 5 akan dijalankan yaitu mid=(left+right)/2 = (1+1)/2 = 1.

		left==right==middle					
	0	1	2	3	4	5	6
data	23	26	38	39	49	90	

Pada pseudo code baris 6, yang terjadi adlh apakah 25 == 26 (ValueToSearch==data[middle]) karena middle bernilai 2 maka yang akan dibandingkan dengan 25 adalah 26 dan hasil perbandingan menghasilkan **false,** sehingga pseudo code baris 7 tidak dijalankan dan pseudo code baris 8 akan dijalankan. Pada pseudo code baris 8, perbandingan yang terjadi

adalah apakah 25<26 dan hasil perbandingan menghasikan **true** sehingga pseudo code baris 9 akan dijalankan dan pseudo code baris 10 dan 11 tidak akan dijalankan sehingga **right** akan bernilai 1-1 = 0 karena digeser menjadi **mid+1** sesuai dengan pseudo code baris 9.

Perulangan berlanjut pada pseudo code baris 4 dimana left<=right menghasilkan **false** karena nilai terakhir left adalah 1 dan nilai terakhir right adalah 0 sehingga 1<0 akan menghasilkan **false**.

Demikian ilustrasi dari semua pseudo code topik pembahasan array. Silahkan implementasikan pseudo code pembahasan array yang sudah dibahas pada ilustrasi sebelumnya pada kegiatan mandiri berikut ini.

Kegiatan Mandiri

NB:

Soal berikut ini boleh dikerjakan dengan data acak atau data yang dimasukkan dari keyboard dengan jumlah data yang ditentukan sendiri.

1. Total & Average
2. Minimum & Maximum
3. Insert and Delete Element Simulation
4. Sequential Search
5. Bubble Sort
6. Insertion Sort
7. Selection Sort
8. Shell Sort
9. Median
10. Binary Search

Bab 7 – Array 2 Dimensi dan Array Dimensi Banyak

Manfaat Pembelajaran:

1. Pengolahan data matrik dan pengolahan citra
2. Artificial Intelligence
3. Computer Vision dan Augmented Reality
4. Pengolahan data vector dan grafika computer
5. Pemrograman game
6. 3D modelling dan Simulation
7. Database design
8. Data mining
9. Data warehouse

Contoh Memasukkan data matrik 2x2 dari keyboard dalam bahasa pemrograman Delphi.

```
Const RowCount:Integer=1;
      ColCount:Integer=1;
Var A:Array[0..RowCount, 0..ColCount]:Integer;
    row, col:Integer;
Begin
 For row:=0 To RowCount Do
  For col:=0 To ColCount Do Begin
   Write('Please enter A[', row, ',',  col, '] : ');
   ReadLn(A[row,col]);
  End;
 Write('These matrix elements have successfully entered:');
 For row:=0 To RowCount Do Begin
  For col:=0 To ColCount Do
   Write(A[row,col], ' ');
  WriteLine;
 End;
```

Hasil tampilan output:

```
Please enter A[0, 0] : 74
Please enter A[0, 1] : 65
Please enter A[1, 0] : 21
Please enter A[1, 1] : 38
These matrikx elements have successfully entered:
74 65
21 38
```

Contoh memasukkan data matrik berukuran 2x2 secara acak dalam bahasa pemrograman Delphi.

```
Uses CRT;
Const RowCount=1;
      ColCount=1;
Var A:Array[0..RowCount, 0..ColCount] : Integer;
    row, col:Integer;
Begin
 Randomize;
 WriteLn('Random generated 2x2 matrix:');
 For row:=0 To RowCount Do Begin
  For col:=0 To ColCount Do Begin
   A[row,col] := Random(100);
   Write(A[row, col]:2, ' ');
  End;
  WriteLn;
 End;
End.
```

Hasil tampilan output:

```
Random generated 2x2 matrix:
 1 46
89 81
```

Kegiatan Mandiri: Buat program untuk menyelesaiakn permasalahan matrik berikut ini berdasarkan pseudocode yang sudah diberikan. Input program boleh dilakukan secara acak ataupun dari keyboard.

Untuk kegiatan mandiri nomor 1 s/d 4 akan menggunakan data matrik 2x3 berikut ini

23 62 12

58 47 85

Dimana:

Matrik[0][0] adalah 23	Matrik[0][1] adalah 62	Matrik[0][2] adalah 12
Matrik[1][0] adalah 58	Matrik[1][1] adalah 47	Matrik[1][2] adalah 85

1. Menjumlahkan sebuah matrik dengan sebuah operand non matrik (variabel atau konstanta).

```
1 INPUT VarOrConst
2 FOR row = 0 TO RowCount - 1
3  FOR col = 0 TO ColCount - 1
4   Matrix[row][col] = Matrix[row][col] + VarOrConst
5  END FOR
6 END FOR
```

Jika nilai **VarOrConst** yang diberikan bernilai **10** maka matrik yang sebelumnya bernilai

23 62 12	Akan menjadi	33 72 22
58 47 85		68 57 95

2. Mengurangkan sebuah matrik dengan sebuah sebuah operand non matrik (variabel atau konstanta).

```
1 INPUT VarOrConst
2 FOR row = 0 TO RowCount - 1
3  FOR col = 0 TO ColCount - 1
4   Matrix[row][col] = Matrix[row][col] - VarOrConst
5  END FOR
6 END FOR
```

Jika nilai **VarOrConst** yang diberikan bernilai **10** maka matrik yang sebelumnya bernilai

23 62 12	Akan menjadi	13 52 2
58 47 85		48 37 75

3. Mengalikan sebuah matrik dengan sebuah sebuah operand non matrik (variabel atau konstanta).

```
1 INPUT VarOrConst
2 FOR row = 0 TO RowCount - 1
3  FOR col = 0 TO ColCount - 1
4   Matrix[row][col] = Matrix[row][col] * VarOrConst
5  END FOR
6 END FOR
```

Jika nilai **VarOrConst** yang diberikan bernilai **10** maka matrik yang sebelumnya bernilai

23 62 12	Akan menjadi	230 620 120
58 47 85		580 470 850

4. Membagi sebuah matrik dengan sebuah sebuah operand non matrik (variabel atau konstanta).

```
1 INPUT VarOrConst
2 FOR row = 0 TO RowCount - 1
3  FOR col = 0 TO ColCount - 1
4   Matrix[row][col] = Matrix[row][col] / VarOrConst
```

```
5   END FOR
6 END FOR
```

Jika nilai **VarOrConst** yang diberikan bernilai **10** maka matrik yang sebelumnya bernilai

23 62 12	Akan menjadi	2 6 1
58 47 85		5 4 8

Untuk kegiatan mandiri nomor 5 dan 6, nilai-nilai matirk A dan B berikut ini akan digunakan sebagai ilustrasi.

Matrik A	Matrik B
23 62 12	8 6 2
58 47 85	5 3 9

5. Menjumlahkan 2 buah matrik berukuran RowCount x ColCount.

```
1 FOR row = 0 TO RowCount - 1
2   FOR col = 0 TO ColCount - 1
3     C[row][col] = A[row][col] + B[row][col]
4   END FOR
5 END FOR
```

Syarat dari operasi penjumlahan matrik adalah jumlah baris dan jumlah kolom dari kedua buah matrik harus sama. Berikut ini ilustrasi penjumalahan matrikA dan B menjadi matrik C.

Matrik A	**+**	**Matrik B**	**=**	**Matrik C**
23 62 12		8 6 2		31 68 14
58 47 85		5 3 9		63 50 94

6. Selisihkan 2 buah matrik berukuran RowCount x ColCount.

```
1 FOR row = 0 TO RowCount - 1
```

```
2  FOR col = 0 TO ColCount – 1
3   C[row][col] = A[row][col] - B[row][col]
4  END FOR
5 END FOR
```

Syarat dari operasi pengurangan matrik adalah jumlah baris dan jumlah kolom dari kedua buah matrik harus sama. Berikut ini ilustrasi selisih matrik A dan B menjadi matrik C.

Matrik A	**-**	**Matrik B**	**=**	**Matrik C**
23 62 12		8 6 2		15 56 10
58 47 85		5 3 9		53 44 76

7. Mengalikan matrik A yang berukuran **ARow x ACol** dengan matrik B yang berukuran **BRow x BCol**.

```
1 FOR row = 0 TO ARow - 1
2  FOR col = 0 TO BCol – 1
3   C[row][col]=0
4   FOR i = 0 TO ACol – 1
5    C[row][col] = C[row][col] + A[row][i]* B[i][col]
6   END FOR
7  END FOR
8 END FOR
```

Syarat dari operasi perkalian matrik adalah jumlah kolom matrik A harus sama dengan jumlah matrik baris B sementara jumlah matrik baris A dan jumlah kolom matrik B tidak menjadi masalah. Berikut ini ilustrasi perkalian matrik A dan B menjadi matrik C.

Matrik A	*****	**Matrik B**	**=**	**Matrik C**
C[0][0] C[0][1]		B[0][0] B[0][1]		C[0][0] C[0][1]
C[1][0] C[1][1]		B[1][0] B[1][1]		C[1][0] C[1][1]

B[2][0] B[2][1]

Jika diuraikan secara lebih terperinci, maka setiap elemen dari matrik C adalah didapatkan dengan rumus:

```
C[0][0] = 0 + A[0][0]*B[0][0] + A[0][1]*B[1][0] + A[0][2]*B[2][0]
C[0][1] = 0 + A[0][0]*B[0][1] + A[0][1]*B[1][1] + A[0][2]*B[2][1]
C[1][0] = 0 + A[1][0]*B[0][0] + A[1][1]*B[1][0] + A[1][2]*B[2][0]
C[1][1] = 0 + A[1][0]*B[0][1] + A[1][1]*B[1][1] + A[1][2]*B[2][1]
-----------------------------------------------------------------
C[b][k] = 0 + A[b][0]*b[0][k] + A[b][1]*b[1][k] + A[b][2]*b[2][k]
```

Penjelasan pseudo code dan proses:

1. **Pseudo code baris 1 dan 2:** Dengan melihat bahwa perkalian matrik A yang berukuran 2x3 dan B yang berukuran matrik 3x2 akan menghasilkan matrik C yang berukuran 2x2, maka dibutuhkan 2 buah perulangan for yaitu for brs dan for klm
2. **Pseudo code baris 3:** Dengan melihat bahwa untuk setiap C, nilai awal sebelum matrik dijumlahkan adalah 0
3. Karena rumus per elemen matrik C yang berhasil didapatkan adalah

   ```
   C[b][k] = 0
    + A[b][0]*b[0][k]    -> A menggunakan kolom 0 dan B menggunakan baris 0
    + A[b][1]*b[1][k]    -> A menggunakan kolom 1 dan B menggunakan baris 1
    + A[b][2]*b[2][k]    -> A menggunakan kolom 2 dan B menggunakan baris 2
   ```

   Maka artinya angka kolom matrik A dan angka baris matrik B adalah berulang dari 0 s/d 2 maka angka 0 s/d 2 menjadi sebuah perulangan I sehingga hal ini sejalan dengan pseudo code baris 4 dan 5

8. Membalikkan matrik berukuran RowCount x ColCount secara horizontal.

```
1 FOR row = 0 TO RowCount - 1
2  i = ColCount - 1
3  FOR col = 0 TO Int(ColCount / 2)
4   Swap(Matrix[row][col], Matrix[row][i])
```

```
5    i = i - 1
6   END FOR
7 END FOR
```

Contoh:

Matrik sebelum dibalikkan	Matrik setelah dibalikkan
23 62 12	12 62 23
58 47 85	85 47 58
34 69 72	72 69 34

Matrik[0][0] akan ditukar dengan matrik[0][2]

Matrik[1][0] akan ditukar dengan matrik[1][2]

Matrik[2][0] akan ditukar dengan matrik[2][2]

9. Membalikkan matrik berukuran RowCount x ColCount secara vertical.

```
1 FOR col = 0 TO ColCount – 1
2   i = RowCount - 1
3   FOR row = 0 TO Int(RowCount / 2)
4    Swap(Matrix[row][col], Matrix[i][col])
5    i = i - 1
6   END FOR
7 END FOR
```

Contoh:

Matrik sebelum dibalikkan	Matrik setelah dibalikkan
23 62 12	34 69 72
58 47 85	58 47 85
34 69 72	23 62 12

Matrik[0][0] akan ditukar dengan matrik[2][0]

Matrik[0][1] akan ditukar dengan matrik[2][1]

Matrik[0][2] akan ditukar dengan matrik[2][2]

10. Memutar matrik SrcMat berukuran SrcHeight x SrcWidth sebesar 90 ^{o} dan simpan hasilnya ke matrik DestMat berukuran DestHeight x DestWidth.

```
1 FOR row0 = 0 TO SrcHeight – 1
2  FOR col0 = 0 TO SrcWidth – 1
3   row = col0
4   col = SrcHeight - row0 - 1
5   DestMat[row][col] = SrcMat[row0][col0]
6  END FOR
7 END FOR
```

Contoh:

Matrik asli sebelum diputar 90^{o}	Matrik hasil setelah diputar 90^{o}
23 62 12	34 58 23
58 47 85	69 47 62
34 69 72	72 85 12

Berikut ini pemetaan koordinat baris dan kolom matrik hasil dari matrik asli (cara baca dari atas ke bawah kemudian kiri ke kanan):

Asli[0][0] -> Hasil[0][2]	Asli[1][0] -> Hasil[0][1]	Asli[2][0] -> Hasil[0][0]
Asli[0][1] -> Hasil[1][2]	Asli[1][1] -> Hasil[1][1]	Asli[2][1] -> Hasil[1][0]
Asli[0][2] -> Hasil[2][2]	Asli[1][2] -> Hasil[2][1]	Asli[2][2] -> Hasil[2][0]

11. Memutar matrik SrcMat berukuran SrcHeight x SrcWidth sebesar 180 ^{o} dan simpan hasilnya ke matrik DestMat berukuran DestHeight x DestWidth.

```
1 FOR row0 = 0 TO SrcHeight – 1
2  FOR col0 = 0 TO SrcWidth – 1
3   row = SrcHeight – row0 – 1
```

```
4    col = SrcWidth – col0 - 1
5    DestMat[row][col] = SrcMat[row0][col0]
6   END FOR
7 END FOR
```

Contoh:

Matrik asli sebelum diputar 180^o	Matrik hasil setelah diputar 180^o
23 62 12	72 69 34
58 47 85	85 47 58
34 69 72	12 62 23

Berikut ini pemetaan koordinat baris dan kolom matrik hasil dari matrik asli (cara baca dari atas ke bawah kemudian kiri ke kanan):

Asli[0][0] -> Hasil[2][2]	Asli[1][0] -> Hasil[1][2]	Asli[2][0] -> Hasil[0][2]
Asli[0][1] -> Hasil[2][1]	Asli[1][1] -> Hasil[1][1]	Asli[2][1] -> Hasil[0][1]
Asli[0][2] -> Hasil[2][0]	Asli[1][2] -> Hasil[1][0]	Asli[2][2] -> Hasil[0][0]

12. Memutar matrik SrcMat berukuran SrcHeight x SrcWidth sebesar 270 ^{o} dan simpan hasilnya ke matrik DestMat berukuran DestHeight x DestWidth.

```
1 FOR row0 = 0 TO SrcHeight – 1
2  FOR col0 = 0 TO SrcWidth – 1
3   row = SrcWidth – col0 - 1
4   col = row0
5   DestMat[row][col] = SrcMat[row0][col0]
6  END FOR
7 END FOR
```

Contoh:

Matrik asli sebelum diputar 270^o	Matrik hasil setelah diputar 270^o
23 62 12	12 85 72
58 47 85	62 47 69
34 69 72	23 58 34

Berikut ini pemetaan koordinat baris dan kolom matrik hasil dari m atrik asli (cara baca dari atas ke bawah kemudian kiri ke kanan):

Asli[0][0] -> Hasil[2][0]	Asli[1][0] -> Hasil[2][1]	Asli[2][0] -> Hasil[2][2]
Asli[0][1] -> Hasil[1][0]	Asli[1][1] -> Hasil[1][1]	Asli[2][1] -> Hasil[1][2]
Asli[0][2] -> Hasil[0][0]	Asli[1][2] -> Hasil[0][1]	Asli[2][2] -> Hasil[0][2]

Bab 8 – Tipe Data String

Manfaat Pembelajaran:

1. Dapat memahami dan mempraktekken pengolahan tipe data kalimat (string) yang dalam prinsipnya memiliki karakteristik yang hampir sama dengan array of char.
2. Mahasiswa mampu mempraktekken konsep pemrograman string dan array pada permasalahan yang ada di lingkungan kita sehari-hari sehingga konsep pemrograman string dan array yang dipelajari memiliki kontribusi dalam kebutuhan sehari-hari.

Berikut ini adalah beberapa contoh pemasalahan komputasi yang melibatkan string dan array dalam penyelesaiannya.

1. Pengolahan bahasa manusia yang digunakan sehari-hari.

   Contoh: memisahkan kalimat “kami sudah mempraktekken algoritma dalam pemrograman komputer hingga bab 7” menjadi array kata yang memisahkan setiap token kata pada kalimat tersebut per elemen kata dalam array dan kemudian mengolah bahasa tersebut lebih lanjut misalnya deteksi kesalahan tata bahasa (grammar) dan memberikan rekomendasi tata bahasa yang benar.
2. Pengolahan bahasa pemrograman tingkat tinggi dalam bidang pengembangan bahasa pemrograman.

   Contoh: pisahkan perintah **Print "Belajar Pemrograman"** menjadi 2 token yang terdiri dari perintah program (yaitu Print) dan parameter perintah (yaitu "Belajar Pemrograman") dan kemudian mengolah bahasa pemrograman tersebut lebih lanjut.
3. Pengolahan ekspresi matematika dalam bidang pengembangan bahasa pemrograman.

   Contoh: memisahkan ekspresi notasi matematika seperti "hasil=3*(a+b)^2-10/6" menjadi sederetan array yang berisi token operand dan operator yang telah dipisahkan dan kemudian mengolahan ekspresi notasi matematika tesebut lebih lanjut.

4. Menerjemahkan nominal uang numerik menjadi bahasa yang berbentuk kalimat yang dibutuhkan dalam bidang keuangan.

   Contoh: menerjemahkan input nominal uang bertipe string seperti "1525500" menjadi "satu juta lima ratus dua puluh lima ribu lima ratus rupiah" dan menampilkan nominal uang tersebut dengan pemisah ribuan sehingga menjadi "1.525.500".

Poin penting perbedaan tipe data string antara Turbo Pascal, Free Pascal dan Delphi

1. Tipe data string Turbo Pascal dan Free Pascal hanya ada 1 yaitu tipe data string yang digunakan untuk menyimpan kalimat yang berisi kode ASCII.
2. Ada 3 tipe data string pada Delphi yaitu ShortString, AnsiString dan UnicodeString. Tipe data ShortString dan AnsiString adalah tipe data string sama persis dimana keduanya menyimpan kalimat yang berisi kode ASCII, keduanya memiliki karakteristik yang sama dan keduanya deprogram dengan cara yang sama. Sedangkan tipe data **String** adalah tipe data string yang dapat menyimpan kalimat yang berisi **Unicode Character** yang dapat menyimpan karakter aksara bahasa di dunia seperti bahasa Mandarin, Jepang, Korea dan bahasa yang menggunakan aksara non-latin.
3. Tipe data string Turbo Pascal dan Free Pascal memiliki panjang yang dapat diatur oleh programmer seperti deklarasi **var s:String[10];** yang artinya variabel s adalah bertipe string yang hanya dapat menampung 10 huruf (ASCII character) dan hal ini tidak dapat dilakukan pada bahasa pemrograman Delphi.

Berikut ini adalah operasi yang dapat dilakukan pada tipe data string dalam bahasa pemrograman Pascal / Delphi.

Operator	Kegunaan	Contoh
:=	Memberikan nilai string dari operand kiri kepada operand kiri	s1:=''; s2:='ini adalah ';
+	Menyambung string, dapat dilakukan di awal string ataupun di akhir string, namun tidak bisa	s1:=s2+'string'; s3:='string '+s2;

	dilakukan di tengah string	
=	Apakah nilai string kiri **sama dengan** string kanan ? Operasi ini akan mengembalikan tipe boolean (true/false)	if s1=s2 then Write('sama');
<>	Apakah nilai string kiri **berbeda dari** string kanan ? Operasi ini akan mengembalikan tipe boolean (true/false)	if s1<>s2 then Write('tidak sama');
<	Apakah nilai string kiri **lebih kecil dari** string kanan ? Operasi ini akan mengembalikan tipe boolean (true/false)	if s1<s2 then Write(lebih kecil');
<=	Apakah nilai string kiri **lebih kecil dari** atau **sama dengan** string kanan ? Operasi ini akan mengembalikan tipe boolean (true/false)	if s1<s2 then Write(lebih kecil atau sama');
>	Apakah nilai string kiri **lebih besar dari** string kanan ? Operasi ini akan mengembalikan tipe boolean (true/false)	if s1>s2 then Write(lebih besar');
>=	Apakah nilai string kiri **lebih besar dari** atau **sama dengan** string kanan ? Operasi ini akan mengembalikan tipe boolean (true/false)	if s1>=s2 then Write(lebih besar');

Kegiatan Mandiri:

1. Buat sebuah program yang dapat menerima input sebuah kalimat kemudian pisahkan menjadi array kata dan tampilkan kembali array kata tersebut dengan karakter spasi (kode ASCII 32) sebagai pemisah kata. NB: Daftar kata harus tersimpan terlebih dahulu ke dalam sebuah array of string sebelum dicetak dengan hasil tampilan berikut ini.

   ```
   Masukkan sebuah kalimat: saya sedang belajar pemrograman C/C++
   Kata-kata yang berhasil dipisahkan:
   saya
   sedang
   ```

```
belajar
pemrograman
C/C++
```

2. Perbaiki kegiatan mandiri nomor 1 untuk mendeteksi dan mengabaikan kelebihan spasi yang terdapat di dalam kalimat yang di-input tersebut, baik kelebihan spasi yang terdapat pada awal kalimat, akhir kalimat ataupun pertengahan kalimat di dalam setiap kata-kata dalam kalimat tersebut
3. Buat sebuah program yang dapat menerima input sebuah ekspresi notasi matematika dan kemudian pisahkan ekspresi notasi matematika tersebut menjadi token-token operand dan operator seperti contoh tampilan di bawah ini. (spasi dalam ekspresi notasi matematika tersebut harus diabaikan).

```
Masukkan sebuah notasi ekspresi matematika: ( x + 2 ) * y
Token yang berhasil dipisahkan:
(
x
+
2
)
*
y
```

4. Buat sebuah program yang dapat menerima input berupa nominal uang (boleh bertipe string ataupun bertipe numerik bulat) dan tampilkan kembali nominal tersebut dalam keadaan sudah terpisah dengan pemisah ribuan.

   NB: Terjemahan ke bahasa boleh ke dalam bahasa Indonesia ataupun bahasa lain.

```
Masukkan sebuah nominal uang: 4580000
Hasil pemisah ribuan: 4.580.000
```

5. Buat sebuah program untuk memasukkan sebuah string dari keyboard lalu membalikkan string tersebut. Sebagai contoh jika string yang dimasukkan adalah **abcde**, maka hasil pembalikkan harus menjadi **edcba**.

6. Buat sebuah program untuk menerjemahkan bilangan desimal menjadi bilangan biner atau bilangan berbasis 2 dengan cara memanfaatkan operator pembagian bilangan bulat / dan operator modulo (sisa bagi) %.

   NB: Sangat disarankan untuk menggunakan tipe data string untuk menyimpan bilangan biner

   Berikut ini adalah contoh proses konversi bilangan desimal menjadi bilangan biner.

   JIka bilangan desimal yang dimasukkan adalah 39 maka lakukanlah proses pembagian dengan bilangan 2 dengan memanfaatkan variabel hasil (nilai awal variabel hasil diambil dari bilangan desimal yang dimasukkan) dan sisa secara berulang-ulang seperti proses berikut ini.

   Lakukan 39 % 2 = **1** (simpan **1** ke variabel biner) dan lakukan 39 / 2 = 19 (nilai 19 akan dibagi lagi)

   Lakukan 19 % 2 = **1** (simpan **1** ke variabel biner) dan lakukan 19 / 2 = 9 (nilai 9 akan dibagi lagi)

   Lakukan 9 % 2 = **1** (simpan **1** ke variabel biner) dan lakukan 9 / 2 = 9 (nilai 4 akan dibagi lagi)

   Lakukan 4 % 2 = **0** (simpan **0** ke variabel biner) dan lakukan 4 / 2 = 2 (nilai 2 akan dibagi lagi)

   Lakukan 2 % 2 = **0** (simpan **0** ke variabel biner) dan lakukan 2 / 2 = 1 (nilai 1 akan dibagi lagi)

   Lakukan 1 % 2 = **1** (simpan **0** ke variabel biner) dan lakukan 1 / 2 = 0 (nilai 0 tidak dibagi lagi, perulangan harus dihentikan)

   Sisa bagi yang sudah disimpan ke variabel biner ini akan menjadi **111001** sedangkan nilai biner sebenarnya dari 39 adalah **100111**. Untuk itu nilai variabel biner **111001** harus dilakukan operasi pembalikan (flip) dengan menambahkan potongan program untuk membalikkan variabel biner.

7. Dengan cara yang sama seperti kegiatan mandiri nomor 6, buat sebuah program untuk menerjemahkan bilangan desimal menjadi bilangan oktal (bilangan berbasis 8)

8. Dengan cara yang sama seperti kegiatan mandiri nomor 6, buat sebuah program untuk menerjemahkan bilangan desimal menjadi bilangan heksadesimal (bilangan berbasis 16) dimana untuk bilangan desimal 10 s/d 15 akan diwakili oleh angka heksa **A** s/d **F**.
   NB: Tidak diperbolehkan menggunakan fitur jadi seperti perintah printf dengan format %x dan %X ataupun fitur lainnya.

9. Buat sebuah program untuk menerjemahkan bilangan biner menjadi bilangan desimal dengan cara mengambil masing-masing digit dari bilangan biner yang tersimpan dalam variabel bertipe string kemudian mengalikan dengan 2 pangkat sekian.
   NB: Boleh menggunakan fungsi hasil=**pow**(n, m); untuk memperoleh nilai n^m dan tambahkan #include **<math.h>** pada bagian atas program.

10. Buat sebuah program untuk menerjemahkan bilangan oktal menjadi bilangan desimal dengan cara yang sama seperti cara mengerjakan kegiatan mandiri nomor 9.

11. Buat sebuah program untuk menerjemahkan bilangan heksadesimal menjadi bilangan desimal dengan cara yang sama seperti cara mengerjakan kegiatan mandiri nomor 9 tetapi membutuhkan sedikit modifikasi terutama pada angka heksa desimal **A** s/d **F**.

Bab 9 – Fungsi Matematika dan Built-in Lainnya

Manfaat Pembelajaran:

1. Menhindarkan pemrogram (programmer) dari kode program numerik dan digital yang rumit akan kebutuhkan fungsi-fungsi matematika yang pada dasarnya tidak tersedia secara hardware.
2. Memudahkan pemrogram dalam memrogram

Pada dasarnya mesin hardware computer hanya bisa mengerjakan 1 operasi dasar (dengan 2 operand) dalam satu satuan waktu dan tidak memiliki fungsi matematika bawaan (built in) hardware komputer. Untuk itu, bahasa pemrograman tingkat tinggi modern sudah menyediakan beberapa fungsi-fungsi matematika yang sebelumnya sudah diprogramkan sedemikian rupa dengan teknik-teknik matematika numerik digital tertentu sehingga pemrogram yang menggunakan bahasa pemrograman modern sudah dapat menikmati fungsi-fungsi matematika ataupun fungsi-fungsi bawaan lainnya untuk memperkaya sebuah bahasa pemrograman. Dengan adanya fungsi-fungsi ataupun fitur-fitur jadi, memrogram sebuah program komputer menjadi sebuah perangkat lunak (software).

Berikut ini adalah beberapa fungsi-fungsi matematika umum dalam bahasa pemrograman Delphi.

1. *ReturnValue*=sqr(Numeric)
   Menghitung dan mengembalikan nilai kuadrat dari sebuah nilai numerik yang dibeirkan
2. *ReturnValue*=sqrt(Numeric)
   Menghitung dan mengembalikan nilai akar kuadrat dari sebuah nilai numeric yang diberikan.
3. *ReturnValue*=exp(Numeric)
   Menghitung dan mengembalikan nilai eksponensial dari sebuah nilai numeric yang diberikan.

4. *ReturnValue*=ln(Numeric)

   Menghitung dan mengembalikan nilai logaritma natural dari sebuah nilai numerik yang diberikan.

5. *ReturnValue*=abs(Numeric)

   Mengembalikan nilai mutlak (non-negatif) dari sebuah numerik yang diberikan dari keyboard.

Berikut ini adalah beberapa fungsi-fungsi trigonometri dan konstanta pi dalam bahasa pemrograman Delphi.

1. *ReturnValue*=sin(Angle_In_Radians)

   Menghitung dan mengembalikan nilai sinus dari sebuah nilai sudut dalam satuan radian.

2. *ReturnValue*=cos(Angle_In_Radians)

   Menghitung dan mengembalikan nilai cosinus dari sebuah nilai sudut dalam satuan radian.

3. pi

   Konstanta ∏ (Pi) dalam bahasa pemrograman Delphi yang bernilai 3.141592653589793.

Berikut ini adalah cara mengkonversi sudut antara satuan radian dan derajat.

```
SudutRadian := SudutDerajat * Pi / 180;

SudutDerajat := SudutRadian * 180 / Pi;
```

Semua fungsi matematika tersebut terdapat di dalam pustaka (library) math dalam bahasa pemrograman Delphi. Apabila dibutuhkan fungsi-fungsi matematika maka diharuskan untuk mengimpor (memasukkan) pustaka math ke dalam program. Berikut ini adalah beberapa cara memasukkan pustaka math dalam bahasa pemrograman Delphi.

```
uses SysUtils;
```

```
var angle:ShortInt;
    radians, s, c, t:Real;

begin
 Write('Please input angle (in degrees): ');
 ReadLn(angle);
 Radians:=angle*Pi/180;
 s:=sin(radians);
 c:=cos(radians);
 t:=tan(radians);
 WriteLn('Sin(' , angle , ')= ' , s);
 WriteLn('Cos(' , angle , ')= ' , c);
 WriteLn('Tan(' , angle , ')= ' , t);
end.
```

Berikut ini adalah beberapa fungsi non matematika yang tersedia dalam bahasa pemrograman Delphi.

1. Val(String, IntOrDecimalNumeric, NumericBase);
   Mengkonversi string menjadi numerik bulat atau numerik pecahan dalam basis bilangan desimal, oktal, biner atau heksadesimal.
2. Str(IntOrDecimalNumeric:x:y, String);
   Mengkonversi numerik bulat atau numerik pecahan dalam format :x:y menjadi string
3. *ReturnValue*=Trunc(DecimalNumericObject)
   Mengambil dan mengembalikan hanya nilai bulat dari sebuah numerik pecahan (membulatkan ke bawah).
4. *ReturnValue*=Frac(DecimalNumericObject)
   Mengambil dan mengembalikan hanya nilai pecahan di belakang pemisah pecahan.
5. *ReturnValue*=chr(IntegerASCIIConstant)
   Mengkonversi sebuah nomor ASCII menjadi sebuah kode ASCII.

6. *ReturnValue*=ord(CharacterConstant)

   Mengkonversi sebuah karakter kode ASCII menjadi nomor ASCII.

7. *ReturnValue*=Round(DecimalNumeric)

   Membulatkan numerik pecahan ke atas.

Kegiatan Mandiri

1. Buat sebuah tabel trigonometri dengan pemrograman Delphi yang menampilkan informasi sudut, sin(sudut), cos(sudut) dan tan(sudut) dimana khusus untuk tangen 90^o dan tangen 270^o harus menampilkan keterangan “tak terhingga”.

   NB:

   Tampilan table trigonometri tidak perlu rapi karena akan dirapikan pada kegiatan mandiri bab User Define Function

2. Buat sebuah tabel ASCII dengan pemrograman Delphi untuk nomor ASCII 30 s/d 122.

Bab 10 – User Define Function

Manfaat Pembelajaran:

1. Mempersingkat program apabila ada beberapa bagian program yang membutuhkan kode program yang sama.
2. Mempermudah perbaikan kesalahan apabila ada beberapa bagian program yang membutuhkan kode program yang sama tersebut ternyata ada yang kurang tepat.
3. Dengan memanfaatkan fungsi maka program menjadi lebih terstruktur dan lebih mudah dipahami.

Selain fungsi-fungsi bawaan (built-in) yang sudah tersedia di dalam setiap bahasa pemrograman, kadang-kadang programmer juga membutuhkan fungsi-fungsi yang tidak terdapat di dalam bahasa pemrograman itu sendiri sehingga programmer harus berinisiatif untuk mengimpor fungsi dari pustaka luar (pihak ketiga) atau membuat sendiri fungsi tersebut. Seperti yang telah dituliskan bahwa manfaat menggunakan function adalah mempersingkat program dan membuat program menjadi lebih terstruktur dan lebih mudah dipahami, maka sebuah fungsi yang dibuat sendiri oleh programmer (user define function) harus efektif dan efisien.

Berikut ini adalah kategori fungsi (sub program).

1. Berdasarkan ketersediaan rancangannya
   a. Fungsi bawaan yang telah disediakan oleh bahasa pemrograman.
   b. Fungsi yang dibuat oleh programmer itu sendiri karena sub program/fungsi tersebut merupakan kebutuhan khusus yang tidak disediakan oleh bahasa pemrograman.
2. Berdasarkan sifatnya

a. Fungsi yang hanya melaksanakan sekumpulan perintah program dan tidak memberikan nilai balik. Ada bahasa pemrograman yang menggunakan sebutan sub atau procedure untuk jenis sub program ini.
b. Fungsi yang setelah melaksanakan sekumpulan perintah program akan mengembalikan sebuah nilai.

Dalam konsep perancangan program, membagi program menjadi beberapa fungsi/sub program adalah disebut sebagai modularisasi (modularizations) yang artinya memecah program menjadi bagian-bagian yang lebih kecil yang memiliki tugasnya masing-masing. Ketika sebuah program dibagi menjadi beberapa modul atau fungsi, program menjadi lebih mudah dirancang dan lebih mudah dirawat/diperbaiki yaitu dengan cara memberikan nama sub program (fungsi) tersebut sesuai dengan fungsi, kegunaan dan tugas daripada sub program/fungsi tersebut agar menjadi lebih mudah diingat oleh programmer. Meskipun program dibagi menjadi sub-sub program (fungsi-fungsi), komputer tetap akan selalu menjalankan perintah program dimulai dari awal program utama hingga akhir daripada program utama.

Sebuah fungsi yang memanggil fungsi yang lainnya biasanya disebut dengan istilah *Calling* Module. Sebuah sub program/fungsi yang dipanggil oleh sub program/fungsi lainnya biasanya disebut dengan istilah *Called Module*.

Berikut ini adalah tahapan merancang sub program yang baik dan benar.

1. Kelompokkan setiap bagian program yang memiliki tujuan yang sama
2. Berikan nama sub program/fungsi untuk setiap kelompok tersebut sesuai dengan kegunaan dari setiap fungsi tersebut
3. Kembangkan/lengkapi isi dari setiap sub program tersebut sesuai dengan nama sub program yang telah diberikan

Berikut ini adalah bentuk umum penulisan fungsi dalam bahasa pemrograman Delphi.

1. Fungsi yang tidak mengembalikan nilai

```
procedure NamaFungsi(VarParam1, VarParam2, ... , VarParamN);
var variabel:TipeData; //deklarasi variabel di sini jika dibutuhkan
begin
 KumpulanPerintahProgram;
end;
```

2. Fungsi yang mengembalikan nilai

```
procedure NamaFungsi(VarParam1,VarParam2,..., VarParamN):TipeData;
var variabel:TipeData; //deklarasi variabel di sini jika dibutuhkan
begin
 KumpulanPerintahProgram;
 NamaFungsi:=hasil;   //atau bisa juga dengan result:=hasil;
end;
```

Perhatikan bahwa VarParam1 s/d VarParam2 wajib diikuti oleh tipe data seperti contoh sub program berikut ini.

Contoh sub program dalam bahasa pemrograman Delphi dan cara pemanggilannya.

```
procedure TambahkanDanCetak(x, y:Integer):Integer;
var z:Integer;
begin
 z := x + y;
 Write(z);
```

```
end;

function Tambahkan(x, y:Integer):Integer;
var z:Integer;
begin
 z := x + y;
 result:=z;
end;

begin   //program utama
 TambahkanDanCetak(5, 10);
 n := Tambahkan(5, 10);
end.
```

Sebagai latihan untuk membuktikan bahwa menggunakan fungsi menyebabkan program menjadi lebih rapi, mudah dipahami dan mudah diperbaiki, perhatikan program menghitung kombinasi dalam bahasa pemrograman Delphi berikut ini.

```
var n, r, FN, FR, FNR, C:Integer;
begin   //program utama
 Write('Please enter value of n: ');
 ReadLn(n);
 Write('Please enter value of r: ');
 ReadLn(r);

 FN:=1;
 for i:=2 to n do FN:=FN*i;

 FR:=1;
 for i:=2 to r do FR:=FN*i;
```

```
 FNR:=1;
 for i:=2 to n-r do FNR:=FNR*i;

 C:=FN/(FNR*FR);
 WriteLn('C(' , n, ',' , r , ') = ', C);
end.
```

Dengan membuat fungsi Faktorial, maka program menjadi seperti berikut ini.

```
function Factorial(n:Integer):Integer;            //The factorial
function
var i, f:Integer;
begin
 f:=1;
 for i:=2 to n do f:=f*i;
 result:=f;
end;

begin   //program utama
 Write('Please enter value of n: ');
 ReadLn(n);
 Write('Please enter value of r: ');
 ReadLn(r);

 C := Factorial(n) / ( Factorial(n-r) * Factorial(r) );
 WriteLn('C(' , n, ',' , r , ') = ', C);
end.
```

Di dalam pemrograman, deklarasi variabel memiliki cakupan global ataupun lokal. Sebuah variabel local yang dideklarasikan di dalam modul (atau fungsi) tersebut hanya dapat dikenal di dalam modul tersebut dan tidak dapat dikenal pada modul lain. Sebuah variabel global adalah sebuah variabel yang sama yang dapat dipanggil pada semua modul. Sebuah variabel global biasanya dideklarasikan pada bagian atas dari semua modul fungsi.

Contoh program penggunaan variabel lokal dan variabel global dalam 1 program yang sama berikut ini.

```
const g:Integer=10;  {this is a global variable}

procedure VariableScopeTestFunction();
var g:Integer;
begin
 g:=20; (*this is still a local variable even if it has the same
     name
          as global variable g*)
end;

{if we declare g variable using var g:Integer here, then this
will be the final version of g variable recognized by main
program}
begin   //main program
 Write(g);   //which variable will be recognized?
end.
```

Pengiriman parameter (passing parameter)

Sebuah parameter di dalam calling module ataupun di dalam sebuah a called module sangat penting dalam mengirimkan data atau informasi yang dibutuhkan function untuk diproses. Biasanya sebuah actual parameter dapat berupa sebuah variabel atau konstanta

sementara sebuah formal parameter wajib berupa variabel. Parameter yang dikirimkan dari calling module kepada called module boleh lebih dari satu parameter dan bahkan fungsi boleh dibuat tanpa menerima parameter apapun.

Berikut ini adalah contoh program dalam bahasa Pascal dengan gaya penulisan Free Pascal yang menunjukkan bahwa program yang menjadi sangat tidak efisien (boros kode program dan tempat penyimpanan) ketika ada beberapa bagian program yang harus dikerjakan oleh program memiliki kode program yang sama persis namun bagian program yang sama tersebut sama sekali tidak menggunakan sub program. Bagi yang menggunakan Turbo Pascal, penulisan function seperti ClrScr(); dan ReadKey(); harus diperbaiki menjadi tanpa tanda ( ) yaitu ClrScr; dan ReadKey;

1. Bagian menu pilih sebagai utama.

```
Uses CRT;

Const max=6;
      m=max-1;

Var data:Array[0..m] of Integer;
    n, i, pos, baru:Integer;
    pil:Char;

Begin
 n:=0;
 Repeat
  ClrScr();
  Write('Data: ');
  For i:=0 To m Do Write(data[i] , ' ');

  WriteLn('1. Sisip Belakang');
  WriteLn('2. Sisip Depan');
  WriteLn('3. Sisip Tengah');
  WriteLn('4. Hapus Belakang');
  WriteLn('5. Hapus Depan');
  WriteLn('6. Hapus Tengah');
  WriteLn('Esc = Keluar');
  Write('Silahkan pilih: ');
  pil:=ReadKey();
  WriteLn(pil);
```

2. Bagian menu 1 s/d 3

```
Case pil of
 '1': If n<max Then Begin
       Write('Masukkan data baru untuk sisip belakang: ');
       ReadLn(baru);
       data[n]:=baru;
       n:=n+1;
      End Else Begin
       WriteLn('Tidak dapat menambah data baru. Kapasitas sudah penuh');
       WriteLn('Silahkan tekan sembarangan tombol untuk kembali ke menu');
       ReadKey();
      End;
 '2': If n<max Then Begin
       Write('Masukkan data baru untuk sisip depan: ');
       ReadLn(baru);
       For i:=n DownTo 1 Do data[i]:=data[i-1];
       data[0]:=baru;
       n:=n+1;
      End Else Begin
       WriteLn('Tidak dapat menambah data baru. Kapasitas sudah penuh');
       WriteLn('Silahkan tekan sembarangan tombol untuk kembali ke menu');
       ReadKey();
      End;

 '3': If n<max Then Begin
       Write('Masukkan data baru untuk sisip tengah: ');
       ReadLn(baru);
       Write('Masukkan posisi sisi yang diinginkan: ');
       ReadLn(pos);
       For i:=n DownTo pos+1 Do data[i]:=data[i-1];
       data[0]:=baru;
       n:=n+1;
      End Else Begin
       WriteLn('Tidak dapat menambah data baru. Kapasitas sudah penuh');
       WriteLn('Silahkan tekan sembarangan tombol untuk kembali ke menu');
       ReadKey();
      End;
```

3. Bagian menu 4 s/d 6 dan akhir program

```
 '4': If n>0 Then n:=n-1 Else Begin
       WriteLn('Tidak ada data yang dapat dihapus');
       WriteLn('Silahkan tekan sembarangan tombol untuk kembali ke menu');
       ReadKey();
      End;
```

```
    '5': If n>0 Then Begin
          For i:=1 To n-1 Do data[i-1]:=data[i];
          n:=n-1
         End Else Begin
          WriteLn('Tidak ada data yang dapat dihapus');
          WriteLn('Silahkan tekan sembarangan tombol untuk kembali ke menu');
          ReadKey();
         End;

    '6': If n>0 Then Begin
          Write('Masukkan posisi hapus yang diinginkan: ');
          ReadLn(pos);
          For i:=pos+1 To n-1 Do data[i-1]:=data[i];
          n:=n-1
         End Else Begin
          WriteLn('Tidak ada data yang dapat dihapus');
          WriteLn('Silahkan tekan sembarangan tombol untuk kembali ke menu');
          ReadKey();
         End;
   End;
  Until pil=#27;
End.
```

Dapat dilihat bahwa kode program pada baris 34 s/d 38 persis digunakan kembali berulang pada (persis sama dengan) baris 46 s/d 50 dan baris 60 s/d 64 (telah ditandai dengan kotak). Begitu juga halnya pada kode program pada baris 67 s/d 70 persis sama dengan kode program pada baris 74 s/d 79 dan baris 86 s/d 90 (telah ditandai dengan kotak) dan jumlah baris kode program adalah 98 baris. Silahkan perhatikan kode program berikut ini yang sudah dimodifikasi dengan memanfaatkan sub program (Perhatikan perubahan pada kode program tersebut dari kode program sebelumnya yang ditandai dengan tanda kotak).

```
Uses CRT;

Const max=6;
      m=max-1;

Procedure TampilPesanPenuh();
Begin
 WriteLn('Tidak dapat menambah data baru. Kapasitas sudah penuh');
 WriteLn('Silahkan menekan sembarangan tombol untuk kembali ke menu');
 ReadKey();
End;

Procedure TampilPesanKosong();
Begin
 WriteLn('Tidak ada data yang dapat dihapus');
 WriteLn('Silahkan menekan sembarangan tombol untuk kembali ke menu');
 ReadKey();
End;

Var data:Array[0..m] of Integer;
    n, i, pos, baru:Integer;
    pil:Char;

Begin
 n:=0;
 Repeat
  ClrScr();
  Write('Data: ');
  For i:=0 To m Do Write(data[i] , ' ');

  WriteLn('1. Sisip Belakang');
  WriteLn('2. Sisip Depan');
  WriteLn('3. Sisip Tengah');
  WriteLn('4. Hapus Belakang');
  WriteLn('5. Hapus Depan');
  WriteLn('6. Hapus Tengah');
  WriteLn('Esc = Keluar');
  Write('Silahkan pilih: ');
  pil:=ReadKey();
  WriteLn(pil);

  Case pil of
   '1': If n<max Then Begin
         Write('Masukkan data baru untuk sisip belakang: ');
         ReadLn(baru);
         data[n]:=baru;
         n:=n+1;
        End Else TampilPesanPenuh();
```

```
    '2': If n<max Then Begin
          Write('Masukkan data baru untuk sisip depan: ');
          ReadLn(baru);
          For i:=n DownTo 1 Do data[i]:=data[i-1];
          data[0]:=baru;
          n:=n+1;
         End Else TampilPesanPenuh();

    '3': If n<max Then Begin
          Write('Masukkan data baru untuk sisip tengah: ');
          ReadLn(baru);
          Write('Masukkan posisi sisi yang diinginkan: ');
          ReadLn(pos);
          For i:=n DownTo pos+1 Do data[i]:=data[i-1];
          data[0]:=baru;
          n:=n+1;
         End Else TampilPesanPenuh();

    '4': If n>0 Then n:=n-1 Else TampilPesanKosong();

    '5': If n>0 Then Begin
          For i:=1 To n-1 Do data[i-1]:=data[i];
          n:=n-1
         End Else TampilPesanKosong();

    '6': If n>0 Then Begin
          Write('Masukkan posisi hapus yang diinginkan: ');
          ReadLn(pos);
          For i:=pos+1 To n-1 Do data[i-1]:=data[i];
          n:=n-1
         End Else TampilPesanKosong();
   End;
 Until pil=#27;
End.
```

Dapat diperhatikan bahwa jumlah baris program setelah menggunakan sub program ada lah 83 baris dimana sebelum menggunakan sub program, jumlah baris kode program adalah 93 baris.

Formal Parameter dan Actual Parameter

Formal parameter adalah parameter yang dideklarasikan dan tersedia di dalam *called module* untuk menerima nilai atau data yang diberikan oleh *calling module*. *Actual parameter* adalah nilai atau data yang dikirimkan oleh *calling module* ketika sebuah *called module* dipanggil.

Variabel yang dideklarasi pada sebuah judul function adalah merupakan formal parameter seperti contoh berikut ini.

```
procedure PrintIfInt(p):
```

Sementara dari contoh pemanggilan di bawah ini, konstanta 10 dan variabel i adalah actual parameter yang dikirimkan kepada fungsi PrintIfInt.

```
begin //program utama
 PrintIfInt(10);
 i:=20;
 PrintIfInt(i);
end.
```

Jika pengiriman parameter diterapkan pada contoh program sisip hapus array yang sebelumnya terhadap fungsi **TampilPesanPenuh()** dan **TampilPesanKosong()** maka kita dapat membuat sebuah fungsi yang lebih universal yang dapat digunakan oleh kedua fungsi tersebut karena mengingat kedua fungsi tersebut hampir sama dan hanya berbeda pada bagian pesan yang dicetak saja.

Berikut ini adalah penulisan kedua sub program tersebut pada program sebelumnya.

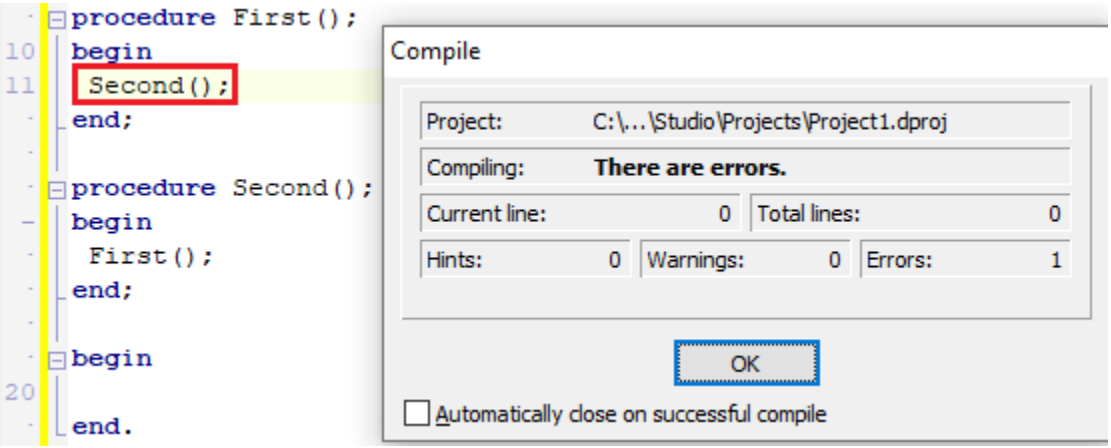
```
Uses CRT;

Const max=6;
      m=max-1;

Procedure TampilPesanPenuh();
Begin
 WriteLn('Tidak dapat menambah data baru. Kapasitas sudah penuh');
 WriteLn('Silahkan menekan sembarangan tombol untuk kembali ke menu');
 ReadKey();
End;

Procedure TampilPesanKosong();
Begin
 WriteLn('Tidak ada data yang dapat dihapus');
 WriteLn('Silahkan menekan sembarangan tombol untuk kembali ke menu');
 ReadKey();
End;

Var data:Array[0..m] of Integer;
    n, i, pos, baru:Integer;
    pil:Char;

Begin
```

Dan berikut ini adalah penulisan kedua sub program tersebut setelah memanfaatkan fungsi TampilPesan() dengan memanfaatkan sebuah parameter bertipe NULL terminated string **pesan** dalam menjalankan menampilkan pesan yang sama tetapi dengan informasi atau pesan yang berbeda.

```
Uses CRT;

Const max=6;
      m=max-1;

Procedure TampilPesan(Pesan:String);
Begin
 WriteLn(Pesan);
 WriteLn('Silahkan menekan sembarangan tombol untuk kembali ke menu');
 ReadKey();
End;

Procedure TampilPesanPenuh();
Begin
 TampilPesan('Tidak dapat menambah data baru. Kapasitas sudah penuh');
End;

Procedure TampilPesanKosong();
Begin
 TampilPesan('Tidak ada data yang dapat dihapus');
End;

Var data:Array[0..m] of Integer;
    n, i, pos, baru:Integer;
    pil:Char;

Begin
```

Dalam bahasa pemrograman Pascal / Delphi, error akan terjadi ketika 2 fungsi atau lebih saling memanggil. Hal ini disebabkan karena bahasa pemrograman Pascal / Delphi adalah bahasa pemrograman yang pada dasarnya bersifat procedural (top down) dimana bahasa pemrograman seperti ini pada dasarnya melakukan single parsing pada saat proses kompilasi dilakukan. Hal ini tentunya berbeda dengan bahasa pemrograman yang sudah full functioning OOP seperti Java dan C# dimana pada bahasa pemrograman yang sudah berbasis full OOP telah melakukan double parsing saat proses kompilasi dilakukan dimana dalam proses kompilasi, parsing yang pertama hanya berfungsi untuk mendata semua dekalrasi tipe data, konstanta, variable dan function dan parsing yang kedua berfungsi untuk memeriksa semua isi dari body function. Contoh program berikut ini akan menyebabkan error pada saat fungsi **First** memanggil fungsi **Second** (error terjadi pada baris ke 3 saat fungsi **First** memanggil fungsi **Second**). Hal ini disebabkan karena pada saat fungsi **First** dideklarasikan, fungsi **Second** masih belum dikenal

(atau belum lahir karena efek daripada single parsing tersebut) dan menghasilkan error **Undeclared Identifier: 'Second'**.



Agar fungsi First dapat memanggil fungsi Second dalam bahasa Pascal / Delphi maka deklarasi fungsi dapat dibuat terlebih dahulu dengan menggunakan keyword **Forward;** sebagai **function interface declaration** sebelum program utama (begin dan end.) kemudian body function dari kedua fungsi tersebut akan dibuat setelah deklarasi forward sebelum program utama seperti contoh program berikut ini.

```
procedure First();  Forward; //tidak perlu dibuat jika tidak dibutuhkan
procedure Second(); Forward;

procedure First();
begin
 Second();
end;

procedure Second();
begin
 First();
end;

begin

end.
```

Catatan Penting:

Dengan mendeklarasikan kedua fungsi tersebut di awal secara forward (tanpa body function) maka pada saat penulisan body function, fungsi **First** mengenal fungsi **Second** dan sebaliknya. Karena method/function **First** dapat memanggil function **Second** tanpa ada masalah dan sebaliknya, maka pesan error bahwa fungsi **Second** tidak dikenal tidak akan terjadi tetapi ada 2

kemungkinan yang terjadi pada saat program dijalankan yaitu program akan menjadi *not responding (hang)* atau pesan error akan dimunculkan karena fungsi yang saling memanggil terus menerus akan menyebabkan memori komputer terus-terusan dipesan untuk mengeksekusi fungsi-fungsi yang saling memanggil tersebut berulang-ualng sehingga ketiga program dijalankan pada sautu waktu tertentu, program akan menghasilkan error dengan pesan Stack Overflow, Out of Memory atau error dengan pesan error yang sejenisnya.

Passing by Value dan Passing by Reference

Perhatikan contoh program berikut ini yang menggunakan pengiriman parameter dengan cara passing by value.

```
Uses CRT;

Procedure Tukar(x, y:Integer);
Var z:Integer;
Begin
 z:=x;
 x:=y;
 y:=z;
End;

Var a, b:Integer;
Begin {Program Utama}
 a:=10;
 b:=20;

 WriteLn('a = ' , a);
 WriteLn('b = ' , b);
 Tukar(a, b); {nilai variabel a diberikan kepada x pada Procedure
 Tukar dan nilai variabel b diberikan kepada variabel y pada baris 9}
 WriteLn('a = ' , a);
 WriteLn('b = ' , b);
End.
```

Perhatikan bahwa variabel x dan y adalah variabel lokal yang hanya dikenal di dalam void Tukar (karena dideklarasikan pada fungsi tukar yang kebetulan dideklarasikan sebagai parameter) dan variabel a dan b adalah variabel lokal yang hanya dikenal di dalam program utama (main) karena didekalrasikan di dalam main.

Saat main memanggil fungsi Tukar(x, y) dengan memberikan nilai variabel a dan b, nilai variabel a digandakan (copy) dan diberikan kepada variabel x dan nilai variabel b digandakan

dan diberikan kepada variabel y sehingga di sini berarti variabel x pada **void Tukar** adalah variabel yang **berbeda** dari variabel a pada **main** dan nama variabel x hanyalah nama variabel yang **mewakili** variabel a di dalam **void Tukar** (karena variabel a tidak dikenal di dalam void Tukar). Demikian halnya juga dengan variabel y pada **void Tukar** adalah variabel yang **berbeda** dari variabel b pada **main** dan nama variabel y hanyalah nama variabel yang **mewakili** variabel b di dalam **void Tukar**.

Hal ini menyebabkan hasil cetakan nilai variabel a dan b pada perintah printf baris 17 setelah pemanggilan void Tukar menjadi sama dengan hasil cetakan variabel a dan b pada baris sebelum pembanggilan void Tukar yaitu pada baris ke 14 sehingga tampilan program akan terlihat seperti berikut:

```
a = 10
b = 20
a = 10
b = 20
```

Agar variabel a dan b tertukar setelah pemanggilan Procedure Tukar maka gunakan keyword var pada deklarasi x dan y untuk melakukan **passing by reference**.

Passing by reference Tambahkan kata kunci var variabel x dan y pada Procedure Tukar.

```
Uses CRT;

Procedure Tukar(Var x, y:Integer);
Var z:Integer;
Begin
 z:=x;
 x:=y;
 y:=z;
End;

Var a, b:Integer;
Begin {Program Utama}
 a:=10;
 b:=20;

 WriteLn('a = ' , a);
 WriteLn('b = ' , b);
 Tukar(a, b); {alamat memori variabel a diberikan kepada x dan
 alamat memori variabel b diberikan kepada y pada baris 9}
 WriteLn('a = ' , a);
 WriteLn('b = ' , b);
End.
```

Kegiatan Mandiri

Perhatikan rancangan algoritma berikut ini untuk mengerjakan kegiatan mandiri nomo 1 s/d 3.

```
Cetak "Please enter a value: "
Masukkan a
Cetak "Please enter b value: "
Masukkan b
c=a+b
Cetak c + "=" + a + "+" + b)
c=a-b
Cetak c + "=" + a + "-" + b)
c=a*b
Cetak c + "=" + a + "*" + b)
c=a/b
Cetak c + "=" + a + "/" + b)
```

1. Perbaiki program utama tersebut menjadi program utama seperti rancangan algoritma berikut ini dan tambahkan sub program yang dibutuhkan

```
Cetak "Please enter a value: "
Masukkan a
Cetak "Please enter b value: "
Masukkan b
c=Tambahkan(a, b)
Cetak c + "=" + a + "+" + b)
c=Kurangkan(a, b)
Cetak c + "=" + a + "-" + b)
c=Kalikan(a, b)
Cetak c + "=" + a + "*" + b)
c=Bagi(a, b)
```

```
Cetak c + "=" + a + "/" + b)
```

2. Perbaiki program utama tersebut menjadi program utama seperti rancangan algoritma berikut ini dan tambahkan sub program yang dibutuhkan

```
Cetak "Please enter a value: "
Masukkan a
Cetak "Please enter b value: "
Masukkan b
c=Tambahkan(a, b)
PrintABC(c, a, "+", b)
c=Kurangkan(a, b)
PrintABC(c, a, "-", b)
c=Kalikan(a, b)
PrintABC(c, a, "*", b)
c=Bagi(a, b)
PrintABC(c, a, "/", b)
```

3. Perbaiki program utama tersebut menjadi program utama seperti rancangan algoritma berikut ini dan tambahkan sub program yang dibutuhkan

```
Cetak "Please enter a value: "
Masukkan a
Cetak "Please enter b value: "
Masukkan b
HitungDanCetakABC(c, a, "+", b)
HitungDanCetakABC(c, a, "-", b)
HitungDanCetakABC(c, a, "*", b)
HitungDanCetakABC(c, a, "/", b)
```

4. Rancang sebuah sub program PrintSpace() untuk melengkapi program berikut ini.

Algoritma ini akan mencetak nilai dengan tampilan rata kiri.

```
For i = 1 To 100
 Print i
End For
```

Perbaiki program di atas menjadi seperti terlihat di bawah ini sehingga variabel i dapat dicetak dengan tampilan rata kanan dengan bantuan fungsi Space() yang telah dirancang

```
For i = 1 To 100
 PrintSpace( 4 - Length( Str(i) ) )
 Print i
End For
```

5. Rancang sebuah sub program Space() untuk melengkapi program berikut ini

Algoritma ini akan mencetak nilai dengan tampilan rata kiri

```
For i = 1 To 100
 Print i
End For
```

Perbaiki program di atas menjadi seperti terlihat di bawah ini sehingga variabel i dapat dicetak dengan tampilan rata kanan dengan bantuan fungsi Space() yang telah dirancang

```
For i = 1 To 100 Do
 Write( Space( 4 - Length( Str(i) ) ), i)
End For
```

6. Rancang sebuah function IsNumeric(String) dalam bahasa pemrograman C/C++ yang dapat mengembalikan nilai True/False apabila semua karakter di dalam string yang diberikan

kepada fungsi IsNumeric merupakan angka kemudian terapkan fungsi tersebut ke dalam pemrograman C/C++.

7. Rancang sebuah function IsUpperCase(String) dalam bahasa pemrograman C/C++ yang dapat mengembalikan status (True/False) apakah semua string yang diberikan adalah huruf besar dan buat juga sebuah function IsLowerCase(String) yang dapat mengembalikan status apakah semua string yang diberikan adalah huruf kecil kemudian terapkan kedua fungsi tersebut ke dalam pemrograman C/C++.
8. Rancang sebuah fungsi IsAlphaNumeric(String) dalam bahasa pemrograman C/C++ apakah sebuah string yang diberikan hanya merupakan huruf besar, huruf kecil, angka dan spasi.
9. Perbaiki program bubble sort, insertion sort, selection sort dan shell sort yang telah dipelajari dengan mengkonversikan program pengurutan tersebut menjadi menggunakan fungsi sesuai dengan kreasi masing-masing.

Bab 11 – Mengatur Tampilan Cetakan dengan Bantuan Fungsi dalam Kasus Perulangan

Manfaat Pembelajaran:

1. Mahasiswa mampu mengatur cetakan tampilan teks (layar console) dengan rapi pada permasalahan komputasi yang melibatkan tampilan table dengan bantuan perulangan.
2. Dengan kemampuan mengatur cetakan tampilan pada layar console sangat membantu mahasiswa dalam mengatur format teks pada operasi file teks yang akan dibahas pada bab selanjutnya.
3. Meningkatkan *sense and feel* mahasiswa akan kemampuan merancang dan mengimplementasikan *user define function* dalam menyelesaikan permasalahan komputasi.

Pada beberapa kasus program yang menggunakan perulangan & percabangan tertentu pengaturan tampilan sangat dibutuhkan. Pengaturan tampilan dibutuhkan untuk membuat beberapa tampilan dalam bentuk tabel ataupun membuat tampilan menjadi lebih rapi.

Mengatur tampilan teks menjadi tercetak rapi dapat dilakukan dengan mudah dengan cara mengatur jumlah spasi yang dibutuhkan untuk mencetak urutan setiap teks tersebut sedangkan untuk mengatur tampilan cetakan numerik bulat membutuhkan proses mengkonversi numerik bulat menjadi string terlebih dahulu kemudian jumlah spasi yang dibutuhkan bias diatur. Sementara itu, mengatur tamplan cetakan numerik pecahan memliki tantangan tersendiri yang lebih rumit untuk jumlah digit pecahan yang harus ditampilkan. Selain itu, fungsi Space(n) akan selalu dibutuhkan untuk mengatur tampilan.

Berikut ini adalah prototype fungsi Space(n) dalam Pseudocode.

```
FUNCTION Space(n)
```

```
 Result=""
 FOR i=1 To n
  Result=Result+" "
 END FOR
 RETURN result
END FUNCTION
```

Perhatikan contoh pengaturan tampilan berikut ini dalam bahasa pemrograman Delphi.

NB:

1. Asumsikan bahwa fungsi space telah dibuat dalam bahasa pemrograman Delphi untuk setiap contoh program berikut ini)NB:
2. Karakter (tanda) '|' sengaja disisipkan untuk melihat apakah hasil tampilan sebelum dan setelah cetakan akan menjadi rapi atau tidak.

1. Mengatur tampilan tabel untuk tipe data string dengan sample data array of string `Nama={'Adi', 'Budi', 'Cindy', 'Edi', 'Fendi', 'Gani', 'Hadi'}`.

   a. Rata kiri

```
for i:=0 to 7 do
 WriteLn(Nama[i], Space(6-Length(Nama[i])), '|');
```

   Hasil tampilan program:

```
Adi    |
Budi   |
Cindy  |
Edi    |
Fendi  |
Gani   |
Hadi   |
```

   Penjelasan:

Jika jumlah huruf terbanyak dari setiap elemen array nama adalah 5 (pada nama Cindy dan Fendi) maka jumlah tempat (karakter) minimum yang dibutuhkan untuk mencetak setiap elemen array nama tersebut adalah 6, sehingga jumlah spasi minimum yang dibutuhkan adalah sebanyak 6-jumlah huruf masing-masing nama. Jika diambil contoh nama Adi dan Fendi, agar cetakan setelah nama Adi dan Fendi menjadi rapi, maka nama Adi membutuhkan 3 spasi (3 huruf + 3 spasi) dan nama Fendi hanya membutuhkan 1 spasi (5 huruf + 1 spasi) sehingga total tempat cetakan yang dibutuhkan tetap 6 dan agar nama tercetak rata kiri, maka spasi harus diberikan setelah cetakan nama.

b. Rata kanan

```
for i:=0 to 7 do
 WriteLn(Space(6-Length(Nama[i])), Nama[i], '|');
```

Hasil tampilan program:

```
   Adi|
  Budi|
 Cindy|
   Edi|
 Fendi|
  Gani|
  Hadi|
```

Penjelasan:

Jika jumlah huruf terbanyak dari setiap elemen array nama adalah 5 (pada nama Cindy dan Fendi) maka jumlah tempat (karakter) minimum yang dibutuhkan untuk mencetak setiap elemen array nama tersebut adalah 6, sehingga jumlah spasi minimum yang dibutuhkan adalah sebanyak 6-jumlah huruf masing-masing nama. Jika diambil contoh nama Adi dan Fendi, agar cetakan setelah nama Adi dan Fendi menjadi rapi, maka nama Adi membutuhkan 3 spasi (3 huruf + 3 spasi) dan nama Fendi hanya membutuhkan 1 spasi (5 huruf + 1 spasi) sehingga total tempat cetakan yang dibutuhkan tetap 6 dan agar nama tercetak rata kiri, maka spasi harus diberikan sebelum cetakan nama.

2. Mengatur tampilan tabel untuk tipe data numerik bulat.

a. Rata kiri

```
n:=5;
j:=1;
for i:=0 to n do begin
 s:=IntToStr(j);
 WriteLn(s, Space(6-Length(s)), '|');
 j:=j*10;
end;
```

Hasil tampilan program:

```
1     |
10    |
100   |
1000  |
10000 |
100000|
```

Penjelasan:

Karena nilai j yang awalnya bernilai 1 terus-menerus dikalikan dengan 10 sebanyak n (5) kali, maka jika nilai j akan dicetak, maka hasil tampilan program yang memungkinkan adalah 1, 10, 100, 100, 1000, 10000 dan 100000. Agar cetakan nilai j tersebut menjadi rata kanan, maka j harus dikonversikan ke tipe data string (variabel s) terlebih dahulu agar fungsi len dapat dipanggil, karena fungsi len umumnya hanya dapat digunakan untuk tipe data string. Spasi sebanyak 6-jumlah huruf variabel s diletakkan setelah mencetak variabel s (yang sebelumnya dikonversi menjadi string dari variabel j) agar cetakan nilai j menjadi rata kanan.

b. Rata kanan

```
n:=5;
j:=1;
for i:=0 to n do begin
```

```
    s:=IntToStr(j);
    WriteLn(Space(6-Length(s)), s, '|')
    j:=j*10;
  end;
```

Hasil tampilan program:

```
     1|
    10|
   100|
  1000|
 10000|
100000|
```

Penjelasan:

Karena nilai j yang awalnya bernilai 1 terus-menerus dikalikan dengan 10 sebanyak n (5) kali, maka jika nilai j akan dicetak, maka hasil tampilan program yang memungkinkan adalah 1, 10, 100, 100, 1000, 10000 dan 100000. Agar cetakan nilai j tersebut menjadi rata kanan, maka j harus dikonversikan ke tipe data string (variabel s) terlebih dahulu agar fungsi len dapat dipanggil, karena fungsi len umumnya hanya dapat digunakan untuk tipe data string. Spasi sebanyak 6-jumlah huruf variabel s diletakkan sebelum mencetak variabel s (yang sebelumnya dikonversi menjadi string dari variabel j) agar cetakan nilai j menjadi rata kanan.

3. Mengatur tampilan tabel untuk tipe data numerik pecahan.
   a. Dalam tampilan biasa (tidak rapi)

```
var i, n:Integer;
    j:Real;
begin
 n:=30;
 for i:=1 to n do begin
  j:=i/3;
  WriteLn(j);
```

```
 end;
end.
```

Hasil tampilan program:

```
3.33333333333333E-0001
6.66666666666667E-0001
1.00000000000000E+0000
1.33333333333333E+0000
1.66666666666667E+0000
2.00000000000000E+0000
2.33333333333333E+0000
2.66666666666667E+0000
3.00000000000000E+0000
3.33333333333333E+0000
3.66666666666667E+0000
4.00000000000000E+0000
4.33333333333333E+0000
4.66666666666667E+0000
5.00000000000000E+0000
5.33333333333333E+0000
5.66666666666667E+0000
6.00000000000000E+0000
6.33333333333333E+0000
6.66666666666667E+0000
7.00000000000000E+0000
7.33333333333333E+0000
7.66666666666667E+0000
8.00000000000000E+0000
8.33333333333333E+0000
8.66666666666667E+0000
9.00000000000000E+0000
9.33333333333333E+0000
9.66666666666667E+0000
1.00000000000000E+0001
```

Kekurangan tampilan:

1. Sulit membedakan antara hasil j yang puluhan dan satuan karena tampilan berbentuk eksponensial.
2. Sangat sulit dipahami orang awam karena ada yang tampil dalam keadaan eksponensial 10 pangkat 1 dan ada yang dalam keadaan eksponensial pangkat 0.

Solusi 1: Gunakan format cetakan :x:y (tidak dibahas lagi karena telah dibahas pada bab 3)

Solusi 2: Gunakan fungsi FormatFloat yang diprogram dengak teknik numerik pecahan dalam pseudo code berikut ini. (NB: Algoritma di dalam fungsi FormatFloat berikut ini bukan merupakan satu-satunya cara)

```
FUNCTION FormatFloat(numeric, nDecimal)
 IF numeric<0 THEN          Jika numeric adalah minus, maka
  Minus="-"                 simpan tanda minus dan operasikan
  numeric=-numeric          numeric sebagai positif
 ELSE                       Jika numeric bernilai positif maka
  Minus=""                  abaikan tanda minus
 END IF

 nZero=1                    Jika nDecimal bernilai 3 maka
 FOR i=1 TO nDecimal        Ulangi i sebanyak 3x agar nZero
  nZero=nZero*10            yang awalnya bernilai 1 menjadi
 END FOR                    10, 100 dan terakhir menjadi 1000
     Jika numeric bernilai 10.33333344 maka tmp akan
 tmp=INT(numeric*nZero)     bernilai 10333 dan frac akan
 frac=STR(INT(tmp) MOD nZero)bernilai STR(10333 MOD 1000) = "333"

 l=len(frac)                Jika jumlah digit pecahan (frac)
adalah
 FOR I=1 TO nDecimal-l      1 maka berikan "0" sebanyak 3-1 = 2
  frac=frac+"0"             (nDecimal- jumlah digit pecahan)
 END FOR                    sehingga frac yg sebelumnya bernilai
                            "0" akan menjadi "000"

 Hasil akhir adalah tanda – digabungkan dengan pembulatan numeric
 sebanyak nDecimal ( INT(tmp/nZero) = 10333/1000 = 10)
digabungkan
 dengan tanda "." dan digabungkan dengan pecahan "333" atau "000"

 result=minus+STR( INT(tmp/nZero) )+"."+frac
 RETURN result              kembalikan format pecahan
END FUNCTION
```

b. Rata kiri

Asumsikan bahwa fungsi Space dan FormatFloat telah dibuat.

```
var i, n:Integer;
    j:Real;
begin
 n:=30;
 for i:=1 to n do begin
  j:=i/3;
  s:=FormatFloat(j, 2);
  WriteLn(s, Space(6-Length(s)), '|');
 end;
end.
```

Hasil tampilan program:

```
0.33  |
0.67  |
1.00  |
1.33  |
1.67  |
2.00  |
2.33  |
2.67  |
3.00  |
3.33  |
3.67  |
4.00  |
4.33  |
4.67  |
5.00  |
5.33  |
5.67  |
6.00  |
6.33  |
6.67  |
7.00  |
7.33  |
7.67  |
8.00  |
8.33  |
8.67  |
9.00  |
9.33  |
9.67  |
10.00 |
```

Perhatikan bahwa cetakan tanda "|" tercetak dengan rapi setelah nilai j meskipun posisi titik (pemisah pecahan) tercetak tidak sejajar.

c. Rata kanan

Asumsikan bahwa fungsi Space dan FormatFloat telah dibuat.

```
var i, n:Integer;
    j:Real;
begin
 n:=30;
 for i:=1 to n do begin
  j:=i/3;
  s:=FormatFloat(j, 2);
  WriteLn(Space(6-Length(s)), s);
 end;
end.
```

Hasil Tampilan program:

```
  2.33
  2.67
  3.00
  3.33
  3.67
  4.00
  4.33
  4.67
  5.00
  5.33
  5.67
  6.00
  6.33
  6.67
  7.00
  7.33
  7.67
  8.00
  8.33
  8.67
  9.00
  9.33
  9.67
 10.00
```

Perhatikan bahwa posisi titik (pemisah pecahan) tercetak dengan rapi (sejajar dan rata kanan), begitu juga dengan tanda "|" yang tercetak setelah nilai j.

Kegiatan Mandiri

1. Buat sebuah program dalam bahasa pemrograman Delphi untuk menampilkan tabel suhu dengan seperti seperti di bawah ini

Celcius	Fahrenheit	Reamur	Kelvin
0	32.00	0.00	273
4	39.20	3.20	277
8	46.40	6.40	281
12	53.60	9.60	285
16	60.80	12.80	289
20	68.00	16.00	293
24	75.20	19.20	297
28	82.40	22.40	301
32	89.60	25.60	305
36	96.80	28.80	309
40	104.00	32.00	313
44	111.20	35.20	317
48	118.40	38.40	321
52	125.60	41.60	325
56	132.80	44.80	329
60	140.00	48.00	333
64	147.20	51.20	337
68	154.40	54.40	341
72	161.60	57.60	345
76	168.80	60.80	349
80	176.00	64.00	353
84	183.20	67.20	357
88	190.40	70.40	361
92	197.60	73.60	365
96	204.80	76.80	369
100	212.00	80.00	373

2. Buat sebuah program dalam bahasa pemrograman Delphi untuk menampilkan tabel trigonometri seperti tampilan seperti di bawah ini. (NB. Gunakan fungsi math.sin(radian), math.cos(radian), math.tan(radian) dan konstanta math.pi atau fungsi math.radians(derajat) untuk menerjemahkan sudut dari satuan derajat menjadi satuan radian)

angle	sin(angle)	cos(angle)	tan(angle)
0	0.000	1.000	0.000
15	0.259	0.966	0.268
30	0.500	0.866	0.577
45	0.707	0.707	1.000
60	0.866	0.500	1.732
75	0.966	0.259	3.732
90	1.000	0.000	Infinity
105	0.966	-0.259	-3.732
120	0.866	-0.500	-1.732
135	0.707	-0.707	-1.000
150	0.500	-0.866	-0.577
165	0.259	-0.966	-0.268
180	0.000	-1.000	-0.000
195	-0.259	-0.966	0.268
210	-0.500	-0.866	0.577
225	-0.707	-0.707	1.000
240	-0.866	-0.500	1.732
255	-0.966	-0.259	3.732
270	-1.000	-0.000	Infinity
285	-0.966	0.259	-3.732
300	-0.866	0.500	-1.732
315	-0.707	0.707	-1.000
330	-0.500	0.866	-0.577
345	-0.259	0.966	-0.268
360	-0.000	1.000	-0.000

Challenges and Excercises

Manfaat Pembelajaran

1. Mengasah kemampuan dalam menyelesaikan permasalahan komputasi, terutama dalam hal pemanfaatan perulangan dan array.
2. Menambah wawasan pembaca mengenai permasalahan di dunia kerja ataupun permasalahan sehari-hari yang dapat dibantu atau diselesaikan dengan pemrograman komputer.

Soal-soal tantangan

1. Buat sebuah program untuk memasukkan sebuah kalimat ke dalam sebuah variabel bertipe string kemudian pisahkan kalimat tersebut menjadi kata-kata per elemen array of string dengan pemisah spasi.

   NB:

   - diharapkan untuk tidak menggunakan fungsi/fitur split bawaan bahasa pemrograman ataupun fitur jadi lainnya yang siap dipake untuk memisahkan kalimat menjadi kata-kata)
   - lebih disarankan untuk menggunakan array dinamis daripada array statis dalam menyelesaikan soal ini

2. Buat sebuah program untuk menerima sebuah persamaan matematika yang terdiri dari banyak operand dan operator dan juga mungkin terdiri dari beberapa tanda ( ) ke dalam sebuah variabel string kemudian pisahkan masing-masing tanda kurung, operand dan operator tersebut menjadi elemen dari sebuah array of string.

   NB:

   - contoh notasi dapat berupa: 5.5*(4-2)^3/4+32 atau a*(b-c)^d/e+32
   - operand dapat berupa variabel ataupun konstanta numerik (bulat ataupun pecahan)

- lebih disarankan untuk menggunakan array dinamis daripada array statis dalam menyelesaikan soal ini

3. Buat sebuah program untuk menghitung nilai trigonometri sin(x) dengan menggunakan deret Taylor/McLaurin berikut ini dimana input sudut harus dalam satuan derajat dan saat menghasilkan nilai sinus dengan deret ini, nilai sudut x harus dalam satuan radian.

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}$$

Soal 4 dan 5 adalah membahas penerjemahan penulisan numerik ratusan menjadi bahasa (dalam kata-kata), silahkan mencoba program ini terlebih dahulu sebelum bisa mengerjakan soal 4 dan 5.

```
uses System.SysUtils;

const Angka:Array[0..9] of ShortString=('', 'satu', 'dua', 'tiga',
       'empat', 'lima', 'enam', 'tujuh', 'delapan', 'sembilan');

var bil, terbilang:ShortString;
    digitkiri, digittengah, digitkanan:SmallInt;

begin
 Write('Masukkan 3 digit bilangan ratusan: ');
 ReadLn(bil);

 terbilang := '';

 digitkiri := Ord(bil[1])-Ord('0');
 terbilang := terbilang + Angka[digitkiri] + ' ratus ';

 digittengah := Ord(bil[2])-Ord('0');
 terbilang := terbilang + Angka[digittengah] + ' puluh ';

 digitkanan := Ord(bil[3])-Ord('0');
 terbilang := terbilang + Angka[digitkanan];

 WriteLn(terbilang);
end.
```

Jika program ini dijalankan:

b. Jika nilai variabel bil yang dimasukkan adalah **234**, maka program akan mencetak output **dua ratus tiga puluh empat**.
c. Jika nilai variabel bil yang dimasukkan adalah **759**, maka program akan mencetak output **tujuh ratus lima puluh sembilan**.
d. Jika nilai variabel bil yang dimasukkan adalah **123**, maka program akan mencetak output **satu ratus dua puluh tiga** dimana hal ini tentunya terasa janggal.
e. Jika nilai variabel bil yang dimasukkan adalah **110**, maka program akan mencetak output **satu ratus satu puluh** dimana hal ini tentunya tidak benar.
f. Jika nilai variabel bil yang dimasukkan adalah **115**, maka program akan mencetak output **satu ratus satu puluh lima** dimana hal ini tentunya tidak benar.
g. Jika nilai variabel bil yang dimasukkan adalah **101**, maka program akan mencetak output **satu ratus puluh satu** dimana hal ini tentunya tidak benar.
h. Jika nilai variabel bil yang dimasukkan adalah **004**, maka program akan mencetak output **ratus puluh empat** dimana hal ini tentunya tidak benar.
i. Jika nilai variabel bil yang dimasukkan adalah **000**, maka program akan mencetak output **ratus puluh** dimana hal ini tentunya tidak benar

Soal-soal tantangan

4. Buat sebuah program untuk memperbaiki program di atas agar masalah pada nomor **c** s/d nomor **h** dapat dicetak oleh program tersebut dengan benar

5. Perbaiki program nomor 3 agar program dapat menerima input bilangan sebesar triliunan dan mencetaknya kembali dalam bahasa (kata-kata)

Soal-soal tantangan yang berhubungan dengan pengolahan kalimat (tipe data string)

6. Buat sebuah program untuk memeriksa apakah sebuah string yang dimasukkan dari keyboard adalah bersifat simetris atau tidak.
7. Buat sebuah program untuk membalikkan semua isi huruf dari sebuah string yang dimasukkan dari keyboard.

8. Buat sebuah program pencarian teks dimana program menerima masukan teks kalimat sebagai target pencarian dan program juga menerima masukan kata yang akan akan dicari apakah terdapat di dalam teks kalimat tersebut atau tidak.

Soal-soal tantangan khusus pengguna Turbo Pascal. Bagi pengguna Free Pascal hal tidak dapat mengerjakan nomor 9 s/d nomor 12. Bagi pengguna Delphi harus mencari dan download Library atau Unit CRT dari internet, letakkan pada 1 folder yang sama dengan project dan tambahkan Uses CRT ke dalam program.

9. Membuat menu utama aplikasi (software) berbasis console

   Tampilan yang diharapkan.

   Cara menggunakan menu yang diharapkan:

   - Jika tombol panah kanan ditekan maka sorotan menu berwarna biru pada menu **Edit** harus bisa berpindah ke menu **Other** dan seterusnya.
   - Jika tombol panah kiri ditekan maka sorotan menu berwarna biru pada menu **Edit** harus bisa berpindah ke menu **File** dan seterusnya.
   - Jika keluar dari program maka tampilan terakhir console (Command Prompt) saat sebelum program dijalankan bisa dikembalikan beserta dengan lokasi kursornya seperti potongan tampilan console berikut.

```
C:\Users\User\Documents>layarteks
C:\Users\User\Documents>_
```

   Ketikkan kode program berikut ini.

```
Uses CRT, DOS;

Const maxConsole=4000;
      nMainMenu=3;
      maxMainMenu=nMainMenu-1;

      MainMenu:Array[0..maxMainMenu] of String[5]=('File', 'Edit', 'Other');
      ScrAddr=$B800;  {Console Screen Memory Address}

Type ConsoleScrType=Array[0..maxConsole-1] of Byte;

Var MainMenuPos:Integer;
    ConsoleScr, LastScr, CurrScr:ConsoleScrType;
    yScr:ShortInt;
    i, j, k:Integer;
    Key:Char;

Begin
 MainMenuPos:=0;
 yScr:=WhereY; {get the current y cursor position}
 If yScr>25 Then yScr:=25; {keep y position to most maximum y}
 Move( Mem[ScrAddr:0], ConsoleScr, maxConsole); {save console screen}

 ClrScr;  {prepare current screen for menu animation}
 i:=0;
 While i<160 Do Begin {fill the first 80 char and color}
  CurrScr[i]:=32; {fill each char byte with a single space}
  i:=i+1;          {switch to color byte}
  CurrScr[i]:=$4F; {4 = red back color, F = white text color}
  i:=i+1;          {switch to the next char byte}
 End;
 While i<4000 Do Begin {the remaining 24x80 char and color}
  CurrScr[i]:=32; {fill each char byte with a single space}
  i:=i+1;          {switch to color byte}
  CurrScr[i]:=$2E; {2 = green back color, E = yellow text color}
  i:=i+1;          {switch to the next char byte}
 End;

 k:=2; {starting to write MainMenu at Console position x=k+1}
 For i:=0 To maxMainMenu Do Begin {for each MainMenu}
  j:=1; {starting from the first char}
  While j<=Length(MainMenu[i]) Do Begin
   CurrScr[k*2]:=Ord(MainMenu[i][j]);
   j:=j+1;
   k:=k+1;
  End;
  k:=k+4;
 End;
```

```
Move( CurrScr, Mem[ScrAddr:0], maxConsole); {put the menu screen}
Move( CurrScr, LastScr, maxConsole); {keep or duplicate current screen data}
TextColor(WHITE);

Repeat
 Move( LastScr, Mem[ScrAddr:0], maxConsole);
 k:=3; {starting to write MainMenu at console position x=3}
 For i:=0 To maxMainMenu Do Begin {for each MainMenu}
  {prepare back color for selected and for non selected}
  If i=MainMenuPos Then TextBackGround(BLUE)
  Else TextBackGround(RED);
  GotoXY(k ,1); {write at position console y=1}
  Write(MainMenu[i]);
  k:=k+8;
 End;

 Key:=ReadKey; {get ASCII keycode}
 If Key=#0 Then Begin {if exteded or arroy button pressed}
  Key:=ReadKey; {get the extended keycode}
  Case Key of
   #75: Begin {if left arrow button pressed, move to}
         MainMenuPos:=MainMenuPos-1; {previous menu circularly}
         If MainMenuPos<0 Then MainMenuPos:=maxMainMenu;
        End;
   #77: Begin
         MainMenuPos:=MainMenuPos+1;
         If MainMenuPos>maxMainMenu Then MainMenuPos:=0;
        End;
  End;
 End;
Until Key=#27;
Move( ConsoleScr, Mem[ScrAddr:0], maxConsole);
GotoXY(1, yScr);
End.
```

10. Sambung kode program pada latihan yang telah dibahas pada nomor 6 untuk isi menu **File** yang lebih kurang terdiri dari **New**, **Open**, **Save**, **Save as** dan **Exit**.

11. Sambung kode program pada latihan yang telah dibahas pada nomor 6 untuk isi menu **Edit** yang lebih kurang terdiri dari **Cut**, **Copy**, **Paste**, **Find** dan **Replace**.

12. Sambung kode program pada latihan yang telah dibahas pada nomor 6 untuk isi menu **Other** yang lebih kurang terdiri dari **Help** dan **About Program**.

Bab 12 – Rekursi

Manfaat Pembelajaran

1. Menyelesaikan permasalahan yang perulangan bolak-balik yang sulit diselesaikan oleh perulangan biasa (looping/iterasi)
2. Menjadi dasar pengetahuan untuk menghasilkan produk-produk program dan perangkat lunak berbasis artificial intelligence.

Rekursi adalah sebuah kondisi dimana sebuah sub program memanggil dirinya sendiri untuk menyelesaikan pekerjaan tertentu yang tidak dapat dikerjakan secara iterasi. Setiap terjadi rekursi pada sebuah sub program, semua nilai variabel dan isi program yang belum dikerjakan pada sub program tersebut akan disimpan ke dalam stack barulah sub program tersebut memanggil dirinya sendiri dengan memesan tempat memori baru untuk menyelesaikan pekerjaan rekursi yang baru sehingga terjadi lompatan antara alamat memori tempat menyimpan nilai variabel dan isi program pada sub program tersebut dengan alamat memori baru tersebut. Setelah sub program rekursi tersebut menyelesaikan pekerjaan atas pemanggilan dirinya sendiri, maka alur program akan kembali kepada sub alamat memori pada stack yang pekerjaannya belum selesai.

Supaya program yang mengandung sub program rekursi tidak mengalami *hang* (not responding) ataupun *stack overflow* (memori stack tidak mencukupi untuk melakukan rekursi) maka sebuah sub program rekursi harus diikuti oleh sebuah perintah percabangan *If*.

Perhatikan contoh program rekursi berikut ini.

Algoritma:

```
Ulang(n)
 Cetak n
 Jika n>1 Maka Panggil Ulang(n-1)
Selesai

Mulai
 Panggil Ulang(5)
Selesai
```

Pseudo code:

```
Function Ulang(n)
 Print n
 If n>1 Then Ulang(n-1)
End Function

Begin
 Call Ulang(5)
End
```

Delphi:

```
procedure Ulang(n:Integer);
begin
 Write(n);
 if n>1 then Ulang(n-1);
end;

begin
Ulang(5);
end.
```

Hasil tampilan program: `54321`

Jika function Ulang(n) tersebut dimodifikasi menjadi berikut ini maka hasil tampilan program akan menjadi `12345`.

```
procedure Ulang(n:Integer);
begin
 if n>1 then Ulang(n-1);
 Write(n);
end;
```

Tetapi jika jika function Ulang(n) tersebut dimodifikasi menjadi berikut ini maka hasil tampilan program akan menjadi `5432112345`.

```
procedure Ulang(n:Integer);
begin
 Write(n);
 if n>1 then Ulang(n-1);
 Write(n);
end;
```

Kegiatan Mandiri

1. Buat program untuk menghasilkan tampilan di bawah ini dengan cara rekursi dengan memanfaatkan dan memperbaiki sub program Ulang pada contoh di atas.

   ```
   12345
   1234
   123
   12
   1
   1
   12
   123
   1234
   12345
   ```

2. Buat program untuk menghitung nilai faktorial dengan menggunakan sub program faktorial rekursi seperti tampilan di bawah ini.

   ```
   Masukkan nilai n: 5
   5! = 120
   ```

3. Buat program untuk menghasilkan nilai kombinasi n terhadap r (nCr) dengan memanfaatkan sub program faktorial yang telah dibuat pada nomor 2. Tampilan program boleh dirancang sendiri.

Bab 13 – Dasar-Dasar Operasi File

Manfaat Pembelajaran:

1. Memahami struktur file (berkas) dan berbagai media penyimpanan agar pengolahan data di dalam file menjadi mudah untuk dilakukan.
2. Program dan perangkat lunak dapat menyimpan berbagai data secara permanen dan universal tanpa bergantung kepada platform OS ataupun bergantung kepada database engine.

Berikut ini adalah berbagai jenis media penyimpanan.

1. Random Access Memory (RAM): EDO-RAM, SD-RAM, DDR-RAM
2. Read Only Memory (ROM): BIOS ROM, Boot ROM
3. Magnetic Disk: Floppy Disk, Tape
4. Non Magnetic Disk: Internal Harddisk, External Harddisk, Flash Disk
5. Fiber Disk: CD-ROM, DVD-ROM, Blue-Ray

Setiap media penyimpanan tidak akan terlepas dari struktur dan sistem penyimpanan dari setiap media. Berikut ini adalah strutur yang paling umum dari semua sistem penyimpanan yang ada pada sitem komputer.

1. Boot Sector
2. File Allocation Table (FAT)
3. Data Area

Berikut ini adalah beberapa jenis (produk) sistem penyimpanan (file system) yang pernah ada.

1. FAT12 & FAT16 (IBM PC-DOS & Microsoft MS-DOS)
2. FAT32, exFAT & NTFS (Microsoft Windows)
3. Linux Native & Linux Swap
4. Ext4 (MacOS)
5. Compact Disk File System (CDFS)

Sebelum dapat mengolah file dengan baik, pembagian jenis-jenis file harus dipahami terlebih dahulu, baik pembagian berdasarkan konten (isi file) dan cara akses, berdasarkan informasi yang terkandung yang di dalamnya, berdasarkan cara akses, berdasarkan format fiel yang telah terdaftar.

Berikut ini adalah pembagian file berdasarkan isi file.

1. File Teks
2. File ASCII (biner)

Berikut ini adalah pembagian file berdasarkan informasi yang terkandung di dalamnya.

1. Text
2. Documents
3. Program Source Code
4. Executable Program Files
5. Script/Batch Files
6. Sketches/Design
7. Image
8. Audio
9. Video
10. Databases

11. Images Files (iso, nrg, mds)
12. Dan lain-lain

Berikut ini adalah pembagian file berdasarkan cara akses.

1. Sequential Access File – Umumnya merupakan file teks
2. Random Access File – Umumnya merupakan file ASCII

Berikut ini adalah pembagian file berdasarkan format file yang dikenal (registered).

1. ws3, ws4, doc, docx
2. wk3, wk4, xls, xlsx
3. ppt, pptx
4. db, dbf, mdb, accdb
5. pcx, pict, bmp, tiff, gif, jpg, png
6. iso, nrg, mds
7. bas, pas, c, cpp, hpp, vb, cs, java, py, html, css, js, php
8. wav, mp3, au, wma
9. avi, mpg, 3gp, vob, wmv, mp4, mkv
10. dan lain-lain

Berikut ini adalah mode akses file yang terdapat di dalam bahasa pemrograman secara umum dan juga terdapat di dalam Delphi.

1. Open
2. Close
3. Read
4. Write
5. Open Mode

Berikut ini adalah perintah dalam bahasa pemrograman Delphi dalam mengoperasikan file.

1. Deklarasi file teks

```
var f:TextFile;
```

2. Deklarasi file biner biasa

```
var f:File;
```

3. Deklarasi file biner terstruktur (record)

```
var f:File of NamaRecordAtauStruct;
```

4. Membuat file teks baru dan membuat file biner terstruktur yang baru

```
{$I-}
Rewrite(f);
{$I+}
```

5. Membuat file biner biasa yang baru

```
{$I-}
Rewrite(f, 1);
{$I+}
```

6. Membuka file teks dan membuka file biner terstruktur untuk dibaca.

```
{$I-}
Reset(f);
{$I+}
```

7. Membuka file biner biasa

```
{$I-}
Reset(f, 1);
{$I+}
```

8. Menambahkan data pada akhir file

```
{$I-}
Append(f);
{$I+}
```

9. Menutup file

```
CloseFile(f);
```

10. Mengambil panjang (ukuran) file

```
Panjang:=FileSize(f);
```

11. Membaca data dari file teks

```
Read(f, var1, var2, ... , varN);
ReadLn(f, var1, var2, ... , varN);
```

12. Menulis data ke dalam file teks

```
Write(f, var1, var2, ... , varN);
WriteLn(f, var1, var2, ... , varN);
```

13. Membaca data dari file biner biasa dan file biner terstruktur

```
BlockRead(f, ArrayOfByteOrCharBuffer, ReadSize);
```

14. Menulis data ke file biner biasa dan file biner terstruktur

```
BlockWrite(f, ArrayOfByteOrCharBuffer, ReadSize);
```

15. Menggeser penunjuk (pointer) file biner.

```
Seek(f, NumberOfBytes);
```

Kegiatan Mandiri

1. Silahkan mencoba program untuk menulis file berikut ini.

```
uses SysUtils;

var f:TextFile;
```

```
    teks:ShortString;

begin
 AssignFile(f, 'textfile.txt');
 {$I-}
 Rewrite(f);
 {$I+}
 teks:='';
 while teks<>'' do begin
  Write('Masukkan sebuah teks: ');
  ReadLn(teks);
  WriteLn(f, teks);
 end;
 CloseFile(f);
end.
```

2. Silahkan mencoba program untuk membaca file berikut ini.

```
uses SysUtils;

var f:TextFile;
    teks:ShortString;

begin
 AssignFile(f, 'textfile.txt');
 {$I-}
 Reset(f);
 {$I+}
 while not EoF(f) do begin
  ReadLn(f, teks);
  WriteLn(teks);
 end;
 CloseFile(f);
end.
```

Bab 14 –Operasi File Lanjutan

Manfaat Pembelajaran:

1. Mampu mengolah file untuk kebutuhan yang lebih kompleks
2. Mampu mengoperasikan dan mengoptimalkan pengolahan data file dengan bantuan list

Berikut ini adalah fitur lanjutan dalam pengolahan file.

1. IntegerReturnValue=FileVar.tell()
   Mengembalikan posisi file pointer (yang bergerak setiap kali membaca atau menulis file) dari sebuah file.
2. FileVar.seek(IntegerPosition, StartLocation)
   Memindahkan posisi file pointer untuk keperluan tertentu dengan mode tertentu.

Mode	Arti
0	Menggeser dari awal file
1	Menggeser dari posisi saat ini
2	Menggeser dari akhir file

Kegiatan Mandiri

1. Rakit sebuah function untuk mengambil ukuran file dengan cara menggunakan fitur tell dan seek.
2. Buat sebuah program untuk menduplikasi file dengan menggunakan fitur read(n) dan write(buffer)

Daftar Pustaka

Cantu, M. (2001). *Mastering Delphi 6.* Sybex.

Mulyono, J. H. (1984). *Teori dan Aplikasi Program Komputer Bahasa Basic.* Yogyakarta: Andi Offset.

Pramono, D. (1999). *Mudah Menguasai C++ Builder Jilid 1.* Jakarta: Elex Media Komputindo.

Pramono, D. (1999). *Mudah Menguasai C++ Builder Jilid 2.* Jakarta: Elex Media Komputindo.

Pramono, D. (1999). *Mudah Menguasai Delphi 3 Jilid 1.* Jakarta: Elex Media Komputindo.

Pramono, D. (1999). *Mudah Menguasai Delphi 3 Jilid 2.* Jakarta: Elex Media Komputindo.

Robertson, L. A. (2006). *Simple Program Design.* Thomson.

Suarga. (2009). *Dasar Pemrograman Komputer dalam Bahasa Java.* Yogyakarta: Andi.

Suarga. (2012). *Algoritma dan Pemrograman.* Yogyakarta: Andi.

Technologies, E. (2021, October 27). *Idera Product Documentation Wiki*. Retrieved from Embarcadero docwiki: https://docwiki.embarcadero.com/

Biodata Penulis 1:
Robin, S.Kom., M.Ti., MTA

Seorang praktisi dan akademisi pada jurusan Teknik Informatika pada salah satu perguruan tinggi swasta di kota Medan. Lahir di kota Medan Sumatera Utara pada tahun 1980 dengan latar belakang pendidikan S1 Teknik Informatika lulusan STMIK Mikroskil Medan dan pendidikan S2 Teknik Informatika lulusan Universitas Bina Nusantara Jakarta.

Pernah 3 tahun berturut-turut mendapatkan penghargaan dosen terbaik pada salah satu perguruan tinggi swasta pada tahun 2014, 2015 dan 2016. Bidang keahlian dan peminatan yang didalami adalah, perancangan infrastruktur pemrograman berbasis pemrograman hardware dengan bahasa assembly, pengolahan citra, grafika komputer dan game engine development. Sebelumnya telah menulis dan menerbitkan buku dengan judul **Mengelola Sumber Daya Sistem Komputer dengan Bahasa Assembly** bersama penulis 2 dan juga telah menerbitkan buku **Algoritma dan Pemrograman Tingkat Dasar** (dalam 3 versi bahasa pemrograman yang berbeda yaitu dalam bahasa pemrograman Delphi, Python dan Java) dan juga telah menerbitkan buku berjudul **Pengembangan Aplikasi Komputer berbasis Cross-Platform Mobile dan Desktop dengan C++ Builder**.

Biodata Penulis 2:
Ali Akbar Lubis, S.Kom., M.TI.

Penulis seorang akademisi yang aktif mengajar di salah satu perguruan tinggi negeri di kota Medan. Penulis menekuni bidang teknik Informatika yang memiliki latar belakang pendidikan Sarjana dari program studi S1 Teknik Informatika, STMIK Mikroskil Medan dan pedidikan magister dari program studi S2 Teknik Informatika, Universitas Bina Nusantara Jakarta.

Penulis lahir di kota Medan, Sumatera Utara pada tahun 1987. Selain mengajar di perguruan tinggi negeri, Penulis juga aktif di beberapa perguruan tinggi swasta di Medan. Peminatan yang didalami oleh penulis adalah bidang pemrograman, pengolahan citra dan pengamanan data.

Biodata Penulis 3:
Irwan Budiman, S.T., M.T.

Seorang akademisi pada Program Studi Teknik Industri di salah satu perguruan tinggi swasta di Medan. Penulis juga merupakan seorang konsultan manajemen pada salah satu perguruan tinggi negeri dan perusahaan Negara maupun daerah. Lahir di Kota Medan dengan latar belakang pendidikan Teknik Industri S1 dan S2, serta memiliki pengalaman kerja di salah satu perusahaan pulp dan paper bertaraf multinasional selama 2012-2015. Dengan pengalaman akademik dan praktisi, penulis berusaha menjembatani kebutuhan dunia bisnis dengan dunia akademisi sehingga memberikan dampak dan berimplikasi bagi dunia bisnis. Bidang keahlian yang didalami terkait dengan Strategic Management, Production Planning & Control, serta Supply Chain Management & Optimization.

ALGORITMA DAN PEMROGRAMAN FREE PASCAL

Perkembangan teknologi hingga saat buku ini ditulis dan dipublikasikan telah mengarah kepada zaman dimana perangkat elektronik, mesin dan komputer telah mulai menggantikan peran sumber daya manusia dalam pekerjaan sehari-hari ataupun profesi tertentu. Beberapa teknologi yang menggantikan peran manusia yaitu akses masuk dengan pengenal wajah, bertransaksi secara digital tanpa harus dilayani pegawai bank, memesan makanan atau layanan transportasi hanya dengan smartphone, pembayaran tagihan lewat layar komputer, pelayanan bandar udara dengan sistem komputer dan masih banyak lagi. Hal ini menyebabkan banyak orang akan kehilangan pekerjaannya karena profesinya telah digantikan oleh komputer, mesin dan perangkat elektronik sehingga buku ini ditulis dengan harapan dapat berkontribusi dalam meningkatkan kualitas generasi muda penerus bangsa dalam hal problem solving dalam dunia digital terutama dalam hal pemrograman dan teknologi perangkat lunak.

Buku algoritma dan dasar pemrograman ini pada awalnya ditulis dalam 4 versi yaitu dalam bahasa pemrograman Python, Delphi dan Java dan C/C++. Sementara buku algoritma dan pemrograman kali ini ditulis dalam Free Pascal.

Buku ini bisa dikategorikan oleh penulis sebagai buku yang "WAJIB BELI" karena buku ini sangat cocok bagi pemula yang menguasasi sedikit ilmu matematika karena buku ini pembahasan buku ini dimulai dari mengimplementasikan metode-metode matematika sederhana ke dalam pemrograman komputer seperti penyelesaian permasalahan dengan rumus-rumus matematika dan fisika, garis bilangan, diagram Venn, sigma, deret dan fungsi hingga kepada kasus komputasi yang lebih mengarah kepada ilmu komputer. Selain itu, buku ini juga menggunakan penjelasan dengan menggunakan gambar ilustrasi, terutama pada topik perulangan, array, matrik dan string.