

# **MODUL PRAKTIKUM ALGORITMA DAN PEMROGRAMAN**



**PROGRAM STUDI D-IV  
POLITEKNIK STATISTIKA STIS**

**2021**



# MODUL

## PRAKTIKUM ALGORITMA DAN PEMROGRAMAN

Disusun oleh :  
Takdir, SST, MT  
Firdaus, MBA  
Ibnu Santoso, SST, MT

Hak Cipta 2021 pada penulis.

Edisi Pertama,  
Cetakan Pertama: 2021

Penerbit:  
Politeknik Statistika STIS  
Jl. Otto Iskandardinata No. 64C Jakarta Timur 13330  
Telpon. (021) 8508812, 8191437,  
Facs (021) 8197577

Hak cipta dilindungi Undang-Undang. Dilarang memperbanyak sebagian atau seluruh isi bahan ajar ini dalam bentuk apa pun, baik secara elektronik maupun mekanik, termasuk mempotokopi, merekap, atau menggunakan sistem penyimpanan lainnya, tanpa izin tertulis dari Penerbit.

UNDANG-UNDANG NOMOR 19 TAHUN 2002 TENTANG HAK CIPTA
<ol style="list-style-type: none"><li>1. <i>Barang siapa dengan sengaja dan tanpa hak mengumukan atau memperbanyak suatu ciptaan atau member izin untuk itu, dipidana dengan penjara paling lama 7 (tujuh) tahun dan atau denda paling banyak Rp. 5.000.000.0000,00 (lima miliar rupiah)</i></li><li>2. <i>Barang siapa dengan sengaja menyiarkan, memamerkan, mengedarkan atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta atau Hak Terkait sebagaimana dimaksud pada ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan atau denda paling banyak Rp. 5.000.000.000,00 (lima miliar rupiah)</i></li></ol>



## KATA PENGANTAR

---

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa karena dengan rahmat, karunia, serta taufik dan hidayah-Nya kami dapat menyelesaikan modul Praktikum Dasar-Dasar Pemrograman ini dengan baik meskipun banyak kekurangan di dalamnya. Kami juga mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu penyusunan modul ini.

Kami sangat berharap modul ini dapat berguna dalam mengantarkan mahasiswa Program D-III Statistika Politeknik Statistika STIS dalam memahami dan menerapkan mata kuliah Komputasi Statistik. Kami juga menyadari sepenuhnya bahwa modul ini terdapat kekurangan dan jauh dari kata sempurna. Oleh sebab itu, kami berharap adanya kritik, saran dan usulan demi perbaikan modul yang telah kami buat di masa yang akan datang, mengingat tidak ada sesuatu yang sempurna tanpa saran yang membangun.

Akhir kata, semoga modul ini dapat bermanfaat dan dipahami bagi siapapun yang membacanya. Tak lupa kami mohon maaf apabila terdapat kesalahan kata-kata yang kurang berkenan dan kami memohon kritik dan saran yang membangun dari pembaca demi perbaikan modul ini di waktu yang akan datang. Terima kasih.

Jakarta, Februari 2021

Tim Penulis



## DAFTAR ISI

---

KATA PENGANTAR.....	iii
DAFTAR ISI.....	v
MODUL 1: Instalasi Compiler, interpreter, dan Software pembuatan Flowchart .....	1
1.1. Deskripsi Singkat .....	1
1.2. Tujuan Praktikum .....	1
1.3. Material Praktikum .....	1
1.4. Kegiatan Praktikum .....	1
A. Instalasi Compiler dan Interpreter.....	1
B. Struktur Bahasa Pemrograman Pascal dan Perintah Standar Output .....	17
C. Software untuk Pembuatan Flowchart .....	18
1.5. Penugasan .....	19
MODUL 2: Pseudocode, Flowchart, Perintah Input, dan Operator Aritmatika .....	20
2.1. Deskripsi Singkat .....	20
2.2. Tujuan Praktikum .....	20
2.3. Material Praktikum .....	20
2.4. Kegiatan Praktikum .....	20
A. Mengamati contoh flowchart dan pseudocode .....	20
B. Perintah Input pada pemrograman Pascal .....	22
C. Operator aritmatik .....	23
2.5. Penugasan .....	24
2.6. Referensi .....	25
MODUL 3: Variabel ,Tipe Data, dan Fungsi Standar Matematika .....	26
3.1. Deskripsi Singkat .....	26
3.2. Tujuan Praktikum .....	26
3.3. Material Praktikum .....	26
3.4. Kegiatan Praktikum .....	26
A. Mencoba berbagai jenis deklarasi variabel dan tipe data .....	26
B. Mencoba Perintah Input dan Format Output pada Variabel.....	27
3.5. Penugasan .....	28
3.6. Referensi .....	29
MODUL 4 STRUKTUR PEMILIHAN/PERCABANGAN IF-THEN .....	30
4.1. Deskripsi Singkat .....	30
4.2. Tujuan Praktikum .....	30

4.3. Material Praktikum.....	30
4.4. Kegiatan Praktikum .....	30
A. IF - THEN .....	30
B. IF-THEN-ELSE .....	31
C. IF Bersarang (Nested IF) .....	33
4.5. Penugasan .....	36
MODUL 5 MULTIPLE SELECTION (STRUKTUR PILIHAN MAJEMUK) .....	37
5.1 Deskripsi Singkat .....	37
5.2 Tujuan Praktikum .....	37
5.3. Material Praktikum.....	37
5.4. Kegiatan Praktikum .....	37
A. Statemen Case .....	37
B. Case bersarang (Nested Case) .....	40
5.5. Penugasan .....	41
MODUL 6 PENGULANGAN 1.....	43
6.1 Deskripsi Singkat .....	43
6.2 Tujuan Praktikum .....	43
6.3 Material Praktikum.....	43
6.4 Kegiatan Praktikum .....	43
A. Struktur FOR.....	43
B. Struktur WHILE-DO.....	46
C. Struktur REPEAT-UNTIL .....	50
D. Pemilihan Struktur yang tepat .....	51
6.5 Penugasan .....	52
MODUL 7 PENGULANGAN 2.....	54
7.1 Deskripsi Singkat .....	54
7.2. Tujuan Praktikum .....	54
7.3. Material Praktikum.....	54
7.4. Kegiatan Praktikum .....	54
A. Statemen BREAK .....	54
B. Statemen CONTINUE.....	55
C. Struktur Pengulangan Bersarang.....	56
7.5 Penugasan .....	59
MODUL 8 TIPE DATA TERBILANG DAN SUB RANGE SERTA SUB PROGRAM 1 .....	60
8.1 Deskripsi Singkat .....	60



8.2 Tujuan Praktikum .....	60
8.3 Material Praktikum .....	60
8.4 Kegiatan Praktikum .....	60
A. Konsep dan Bentuk Umum Tipe Data Terbilang dan Sub Range .....	60
B. Konsep dan Bentuk Umum Fungsi dan Prosedur .....	62
8.5 Catatan: .....	65
8.6 Penugasan .....	65
MODUL 9 SUB PROGRAM 2 .....	66
9.1 Deskripsi Singkat .....	66
9.2 Tujuan Praktikum .....	66
9.3 Material Praktikum .....	66
9.4 Kegiatan Praktikum .....	66
A. Identifier .....	66
B. Transfer Parameter .....	72
9.5 Penugasan .....	73
MODUL 10 ARRAY .....	75
10.1 Deskripsi Singkat .....	75
10.2 Tujuan Praktikum .....	75
10.3 Material Praktikum .....	75
10.4 Kegiatan Praktikum .....	75
A. Array 2 Dimensi .....	79
C. Array Multi Dimensi .....	81
10.5 Penugasan .....	82
MODUL 11 RECORD .....	84
11.1 Deskripsi Singkat .....	84
11.2 Tujuan Praktikum .....	84
11.3 Material Praktikum .....	84
11.4 Kegiatan Praktikum .....	84
11.5 Penugasan .....	90
MODUL 12 REKURSI .....	92
12.1 Deskripsi Singkat .....	92
12.2 Tujuan Praktikum .....	92
12.3 Material Praktikum .....	92
12.4 Kegiatan Praktikum .....	92
A. Fungsi Rekursif .....	92

B. Prosedur Rekursif .....	95
C. Rekursi Tak Hingga .....	96
D. Special Case pada Rekursi .....	97
E. Merancang Algoritma Rekursi .....	98
F. Fungsi Rekursif atau Prosedur Rekursif .....	99
12.5 Penugasan .....	100
MODUL 13 PENCARIAN .....	102
13.1 Deskripsi Singkat .....	102
13.2 Tujuan Praktikum .....	102
13.3 Material Praktikum .....	102
13.4 Kegiatan Praktikum .....	102
A. Pencarian Sekuensial pada Array Tidak Terurut .....	102
B. Pencarian Sekuensial pada Array Terurut .....	103
C. Pencarian Biner .....	104
13.5. Penugasan .....	106
MODUL 14 PENGURUTAN .....	108
14.1 Deskripsi Singkat .....	108
14.2 Tujuan Praktikum .....	108
14.3 Material Praktikum .....	108
14.4 Kegiatan Praktikum .....	108
A. Bubble Sort .....	108
B. Selection sort .....	109
C. Insertion Sort .....	109
14.5 Penugasan .....	112

## MODUL 1: Instalasi Compiler, interpreter, dan Software pembuatan Flowchart

---

### Deskripsi Singkat

Praktikum ini merupakan langkah awal untuk mengenal bahasa pemrograman dan menggunakan software kompil器和 interpreter untuk membuat dan menjalankan program, serta penggunaan software untuk membantu pembuatan flowchart.

### Tujuan Praktikum

Setelah praktikum pada modul 1 ini diharapkan:

1. Mahasiswa mampu melakukan instalasi compiler dan interpreter untuk melakukan pemrograman Pascal
2. Mahasiswa memahami struktur bahasa pemrograman pascal dan memahami perintah standard output
3. Mahasiswa mampu mengoperasikan software untuk membuat flowchart

### Material Praktikum

Material praktikum ini terdiri dari file instalasi yang dapat diunduh dari internet dengan link yang dicantumkan pada modul.

### Kegiatan Praktikum

#### Instalasi Compiler dan Interpreter

Ada berbagai jenis compiler dan interpreter yang dapat digunakan untuk melakukan pemrograman Pascal, diantaranya adalah:

**Turbo Pascal** – compiler dan interpreter yang berbasis text user interface dan dapat berjalan pada sistem operasi DOS, Windows, dan Machintos.

**Delphi** – compiler dan interpreter Object Pascal yang dapat meng-generate kode native untuk Windows, serta Mac OS X dan iOS.

**Free Pascal.** Compiler yang menyediakan fitur kompilasi program Pacal untuk berbagai platform dan sistem operasi yang berbasis kompil器和 Turbo Pascal dan Delphi.

**Turbo51** – compiler Pascal untuk pemrograman microcontrollers.

**Oxygene** – compiler Pascal untuk .NET and Mono platforms.

**GNU Pascal (GPC)** – compiler Pascal yang tergabung dalam GNU Compiler Collection yang merupakan tools yang menyediakan software untuk kompilasi untuk berbagai jenis bahasa pemrograman.

Pada praktikum ini akan digunakan Free Pascal sebagai compiler. Untuk tahapan instalasi, silakan mengikuti instruksi yang disediakan pada link berikut:

Unduh compiler Free Pascal pada link berikut sesuai dengan jenis sistem operasi yang digunakan: <https://www.freepascal.org/download.var>

Lakukan instalasi dengan mengikuti petunjuk berikut

Linux: lihat petunjuk pada <https://www.freepascal.org/docs-html/user/usersu5.html>

Mac: Install Xcode dari Apple App Store dan ikuti instruksi instalasi. Setelah itu, Xcode dapat digunakan untuk menjalankan compiler Free Pascal

Windows: Ikuti petunjuk pada <https://www.freepascal.org/docs-html/user/usersu3.html>

Berikut contoh langkah-langkah dan tampilan untuk instalasi pada sistem operasi Windows:

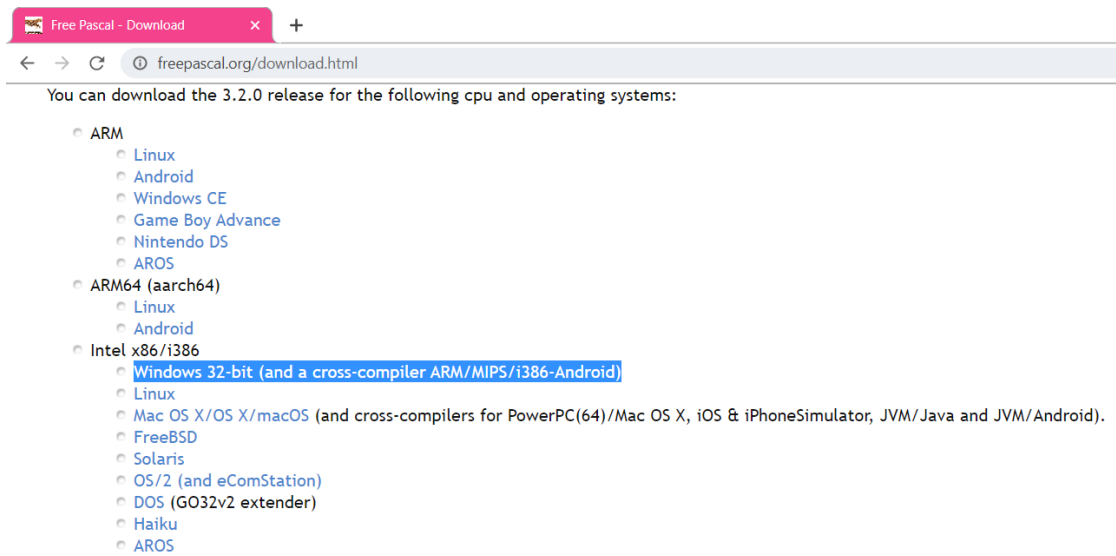
### **i) Instalasi Compiler**

Langkah pertama yang harus dilakukan untuk menyiapkan laptop/komputer kita agar siap dalam melakukan pemrograman Free Pascal adalah melakukan download file instalasi pada halaman resminya di link dibawah ini:

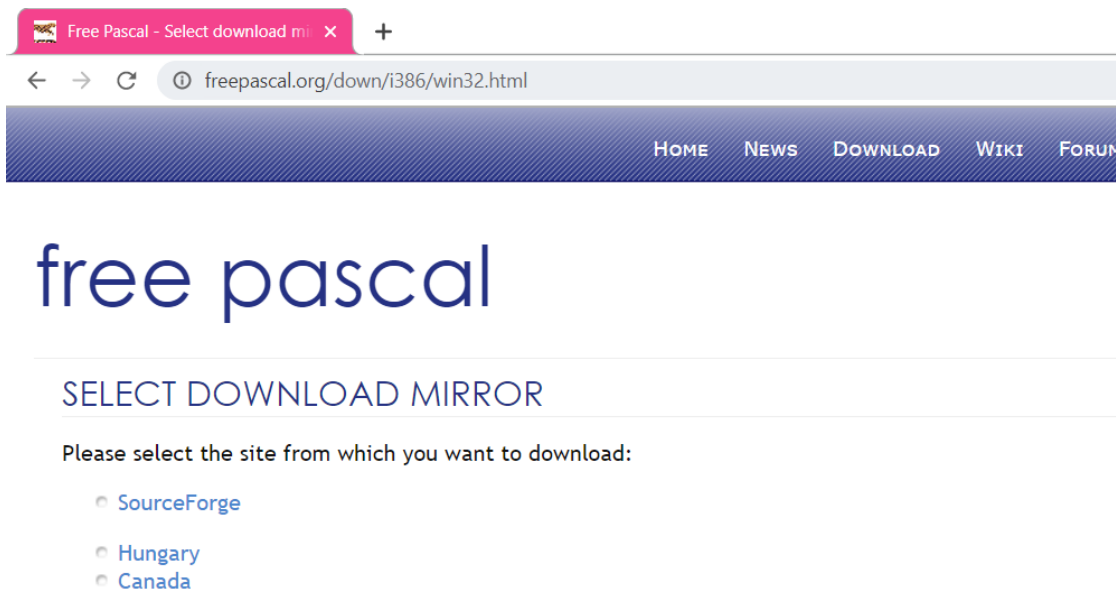
halaman resminya di link dibawah ini:

<https://www.freepascal.org/download.html>

Pada halaman tersebut silahkan pilih versi sistem operasi yang anda gunakan. Misalnya kita menggunakan Windows maka klik pada Windows 32-bit.



Kemudian pilih server untuk download.



Misalkan kita pilih *SourceForge*.

Free Pascal - Select download mirror

freepascal.org/down/i386/win32.html

HOME NEWS DOWNLOAD WIKI

# free pascal

## SELECT DOWNLOAD MIRROR

Please select the site from which you want to download:

- ☒ SourceForge
- ☐ Hungary
- ☐ Canada

Halaman akan dialihkan ke halaman *SourceForge* untuk memilih versi *release*.

Free Pascal - Select download mirror | Free Pascal Compiler - Browse / Files

sourceforge.net/projects/freepascal/files/Win32/3.2.0/

SOURCEFORGE

Open Source Software Business Software Resources

Home / Browse / Development / Compilers / Free Pascal Compiler / Files

## Free Pascal Compiler

Free 32/64/16-bit multi-platform Pascal and Object Pascal compiler  
Brought to you by: dmanitone, jmaebe, loesje, marcovtje, and 6 others

Summary Files Reviews Support Wiki Code News Discussion

Download Latest Version  
fpc-3.2.0.i386-win32.exe (53.1 MB) Get Updates

Home / Win32 / 3.2.0

Name	Modified	Size	Downloads / Week
Parent folder			
readme.txt	2020-06-16	16.1 kB	0
fpc-3.2.0.i386-win32.cross.i8086-msdos.exe	2020-06-16	168.2 MB	57
fpc-3.2.0.i386-win32.exe	2020-06-16	53.1 MB	4,395
fpc-3.2.0.i386-win32.cross.x86_64-win64.exe	2020-06-16	40.8 MB	406
fpc-3.2.0.i386-win32.cross.arm-wince.exe	2020-06-16	27.6 MB	13
fpc-3.2.0.i386-win32.cross.android.exe	2020-06-16	99.4 MB	23

Pilih *fpc-3.2.0.i386-win32.exe*

Free Pascal - Select download mi x Free Pascal Compiler - Browse /v x +

sourceforge.net/projects/freepascal/files/Win32/3.2.0/

**SOURCEFORGE**

Open Source Software Business Software Resources

Home / Browse / Development / Compilers / Free Pascal Compiler / Files

# Free Pascal Compiler

Free 32/64/16-bit multi-platform Pascal and Object Pascal compiler  
Brought to you by: [dmantione](#), [jmaebe](#), [loesje](#), [marcovtje](#), and 6 others

Summary **Files** Reviews Support Wiki Code News Discussion

**Download Latest Version**  
fpc-3.2.0.i386-win32.exe (53.1 MB) **Get Updates**

Home / Win32 / 3.2.0

Name	Modified	Size	Downloads / Week
<a href="#">Parent folder</a>			
<a href="#">readme.txt</a>	2020-06-16	16.1 kB	57
<a href="#">fpc-3.2.0.i386-win32.cross.i8086-msdos.exe</a>	2020-06-16	168.2 MB	4,395
<b><a href="#">fpc-3.2.0.i386-win32.exe</a></b>	2020-06-16	53.1 MB	406
<a href="#">fpc-3.2.0.i386-win32.cross.x86_64-win64.exe</a>	2020-06-16	40.8 MB	13
<a href="#">fpc-3.2.0.i386-win32.cross.arm-wince.exe</a>	2020-06-16	27.6 MB	

Kemudian tunggu proses..

Free Pascal - Select download mi x Download Free Pascal Compiler x +

sourceforge.net/projects/freepascal/files/Win32/3.2.0/fpc-3.2.0.i386-win32.exe/download

**SOURCEFORGE**

Open Source Software Business Software Resources

Home / Browse / Development / Compilers / Free Pascal Compiler

# Free Pascal Compiler

Free 32/64/16-bit multi-platform Pascal and Object Pascal compiler  
Brought to you by: [dmantione](#), [jmaebe](#), [loesje](#), [marcovtje](#), and 6 others

Your download will start shortly... 2

**Get Updates** **Share This** **Problems Downloading?**

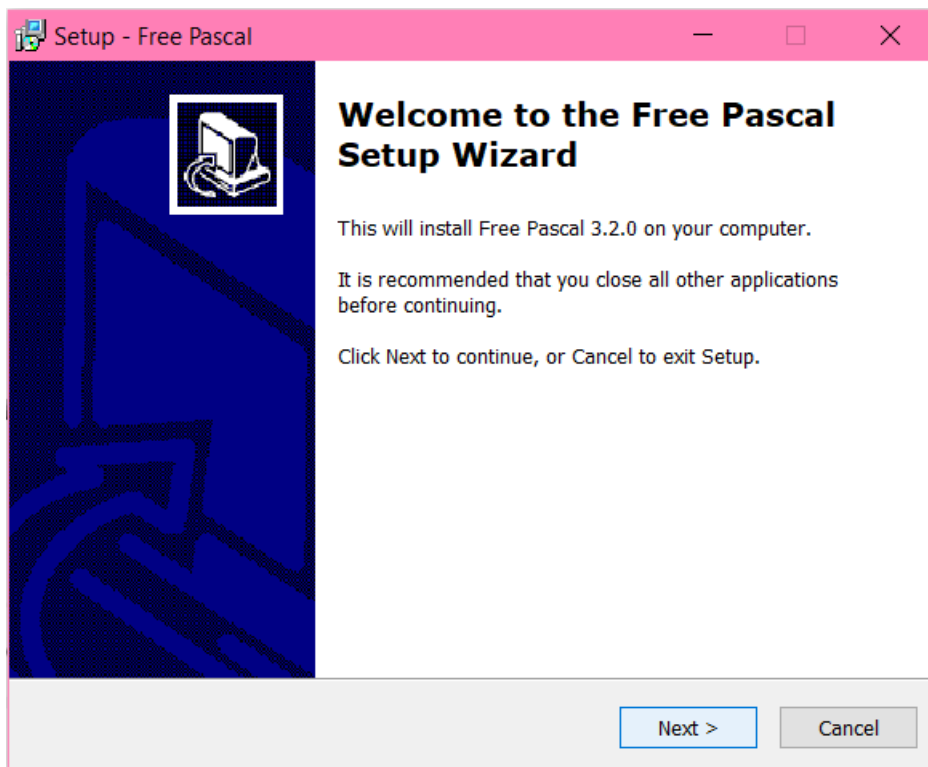
fpc-3.2.0.i386-win32.exe | Scanned for malware ✓

Kemudian tunggu proses download selesai.

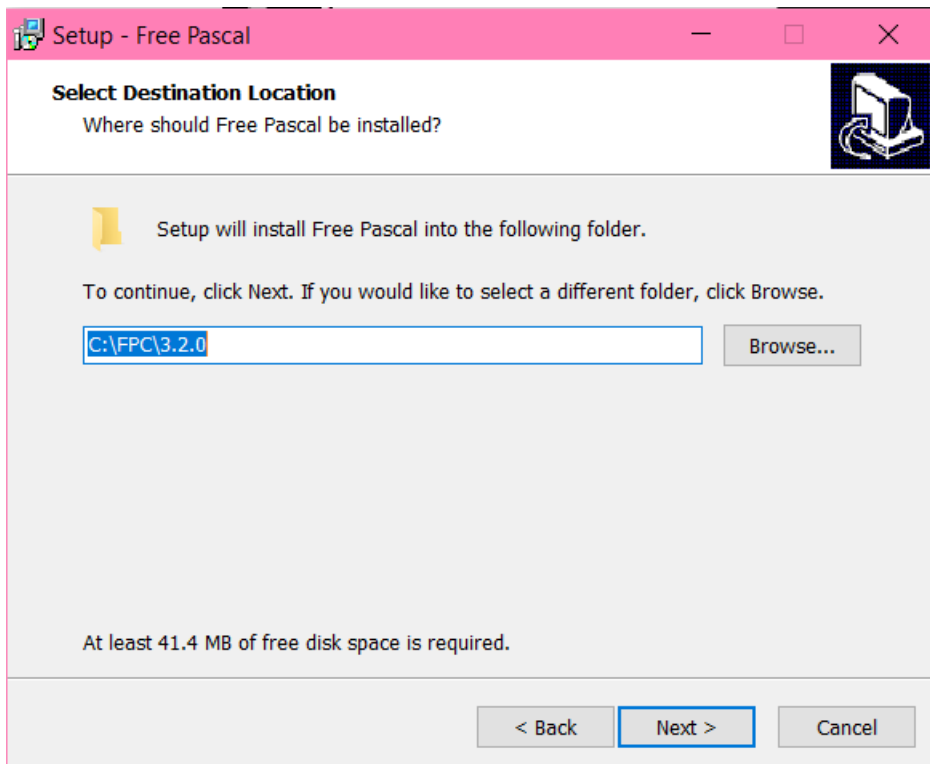
fpc-3.2.0.i386-win....exe  
1.1/50.7 MB, 2 mins left

Jalankan file instalasi.

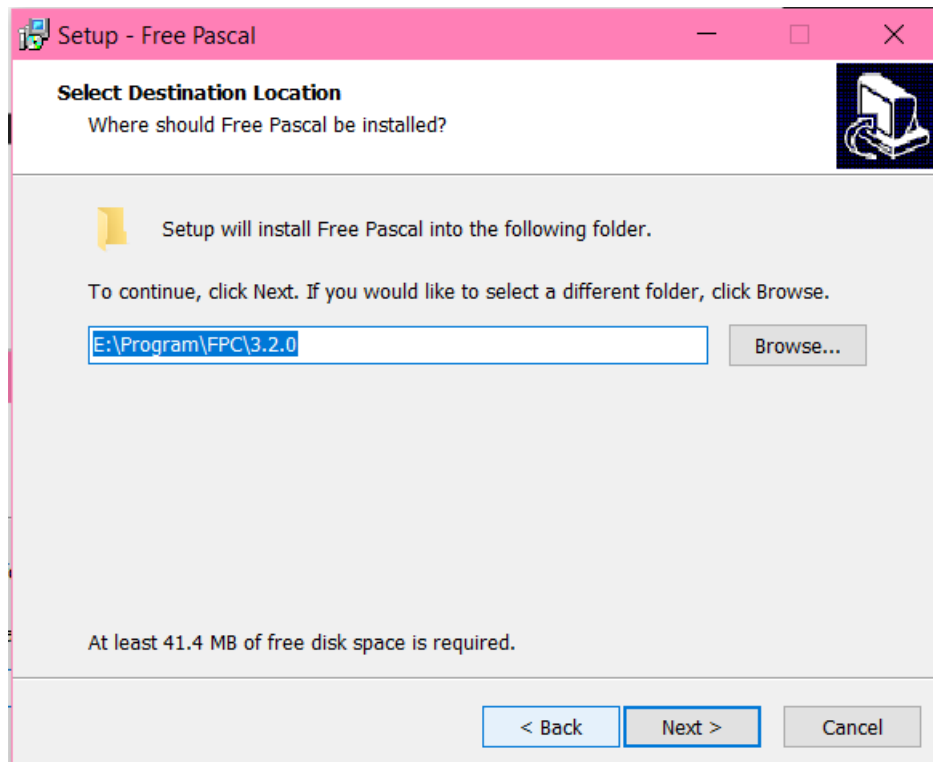
Klik next.



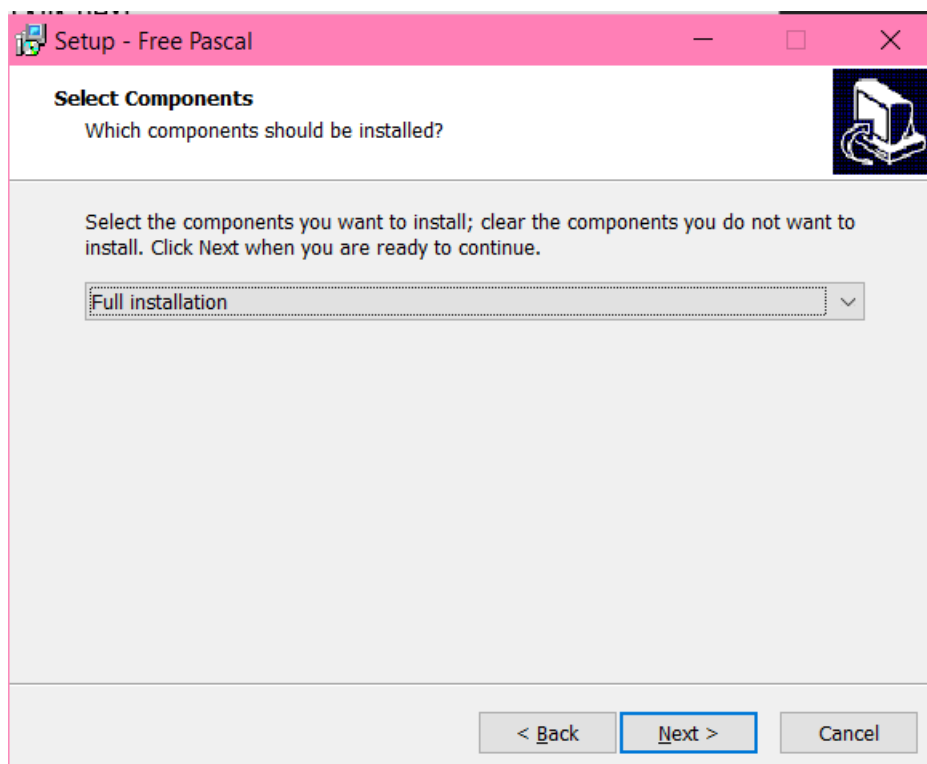
Tentukan lokasi file instalasi. (bisa diketikkan atau browse untuk memilih lokasi)



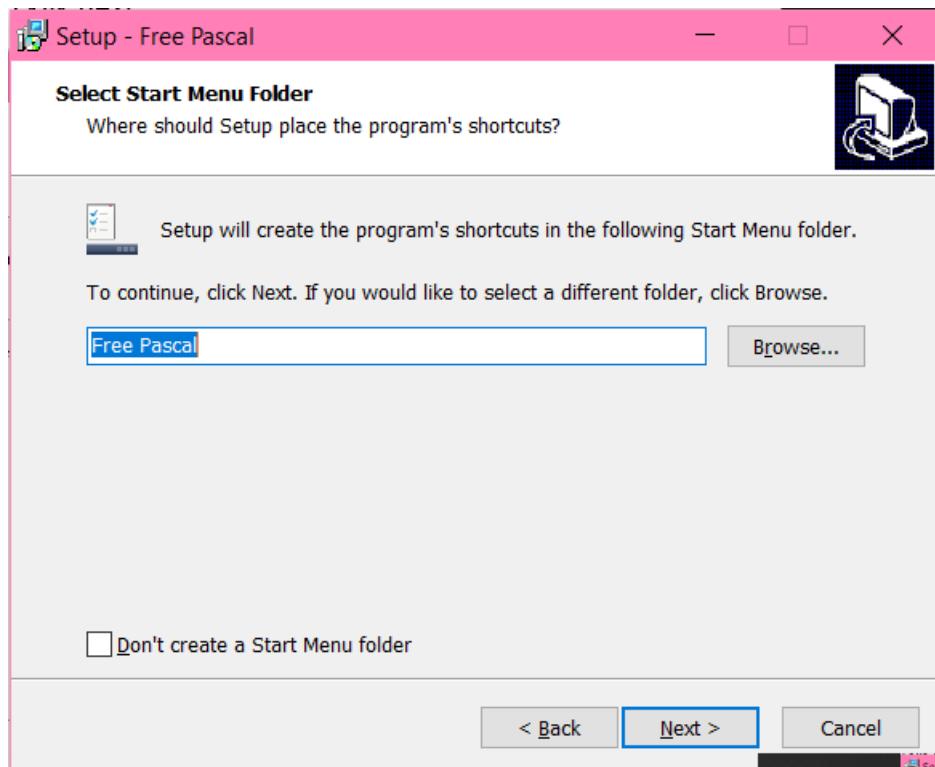




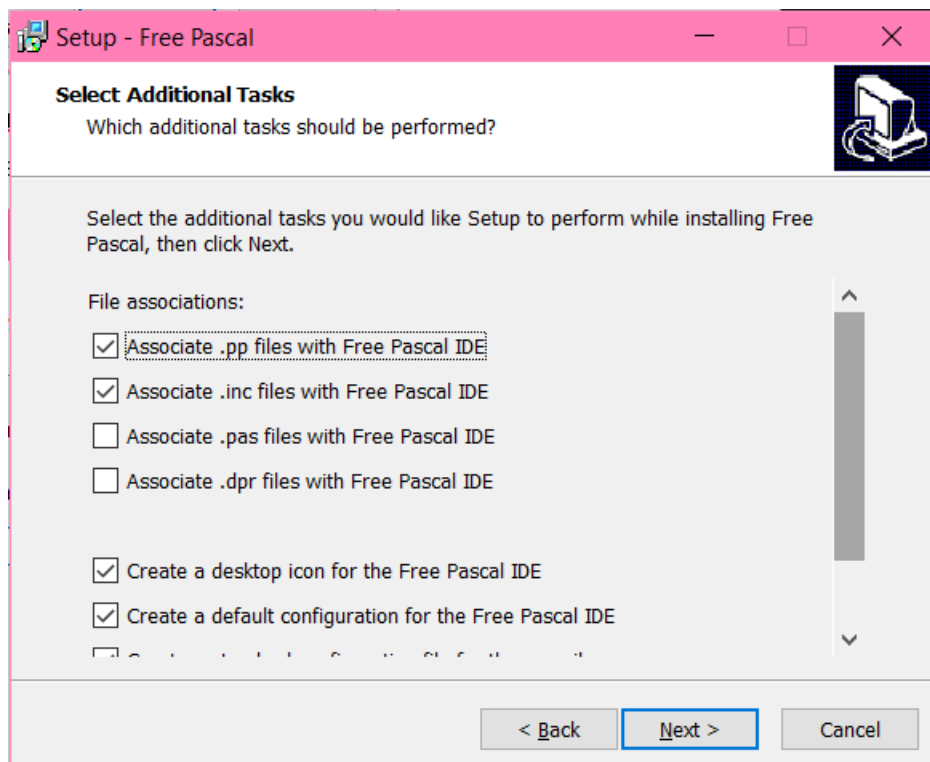
Pilih tipe instalasi, misalkan *full installation*, dan klik next

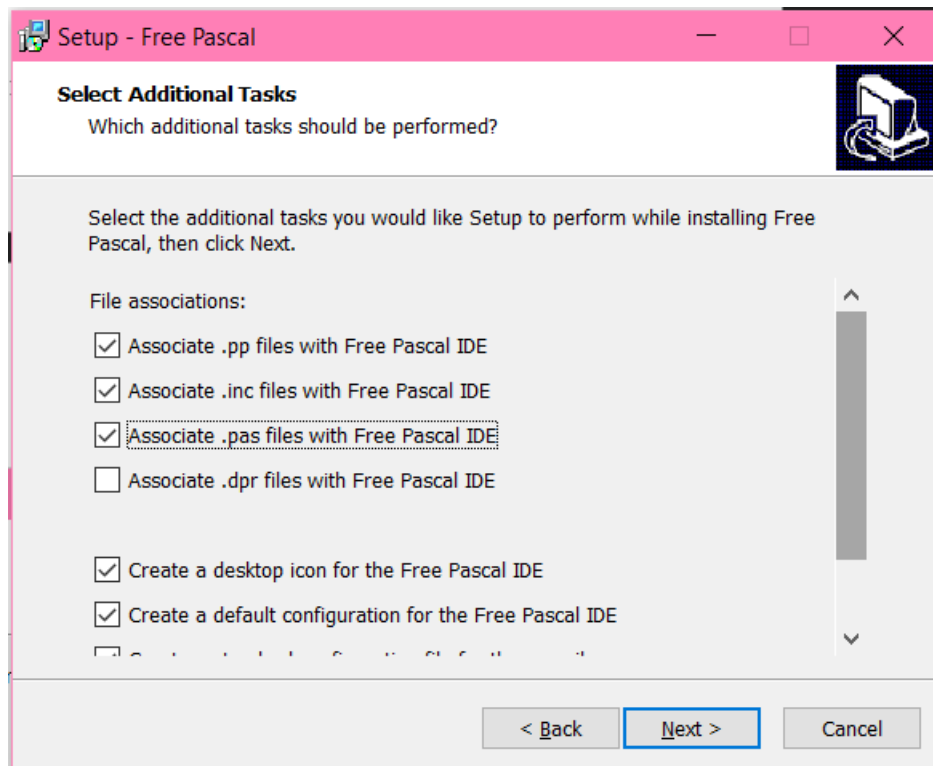


Pilih lokasi shortcut akan diletakkan. Jika ingin mengubah, klik browse dan pilih lokasi. Jika tidak ingin mengubah langsung klik next.

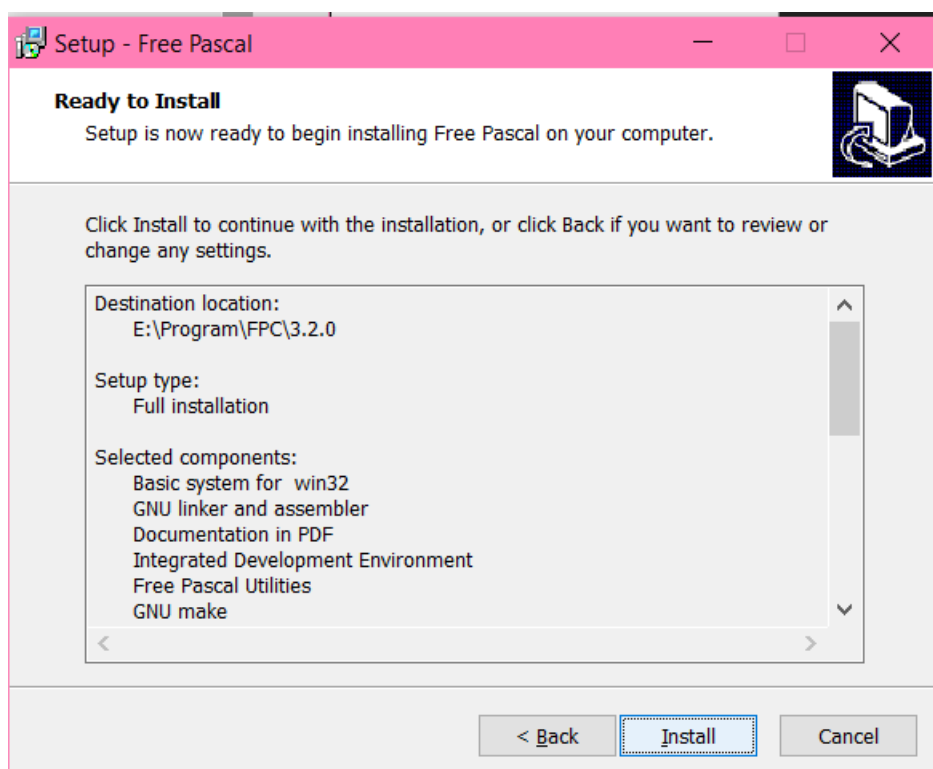


Pilih ekstensi file yang akan diasosiasikan dengan software free Pascal.

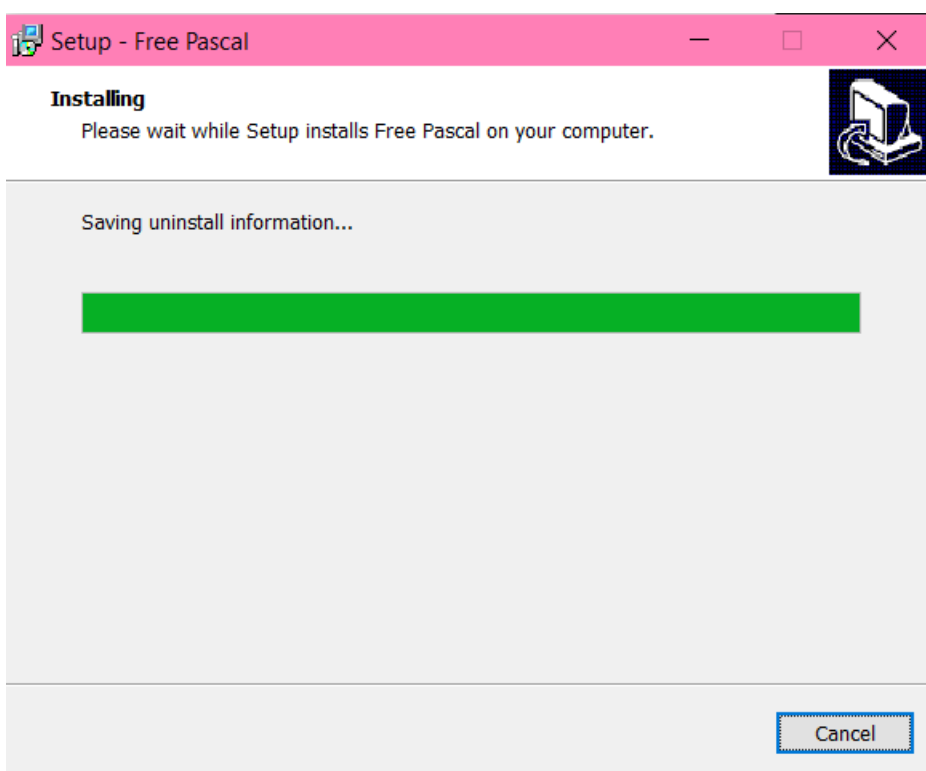
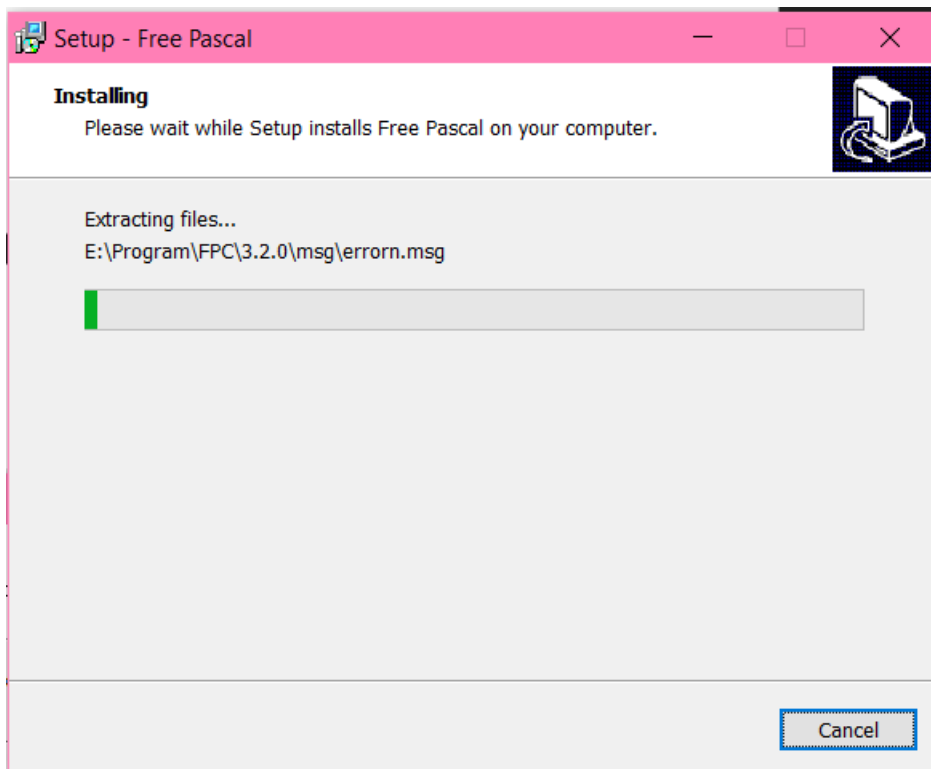




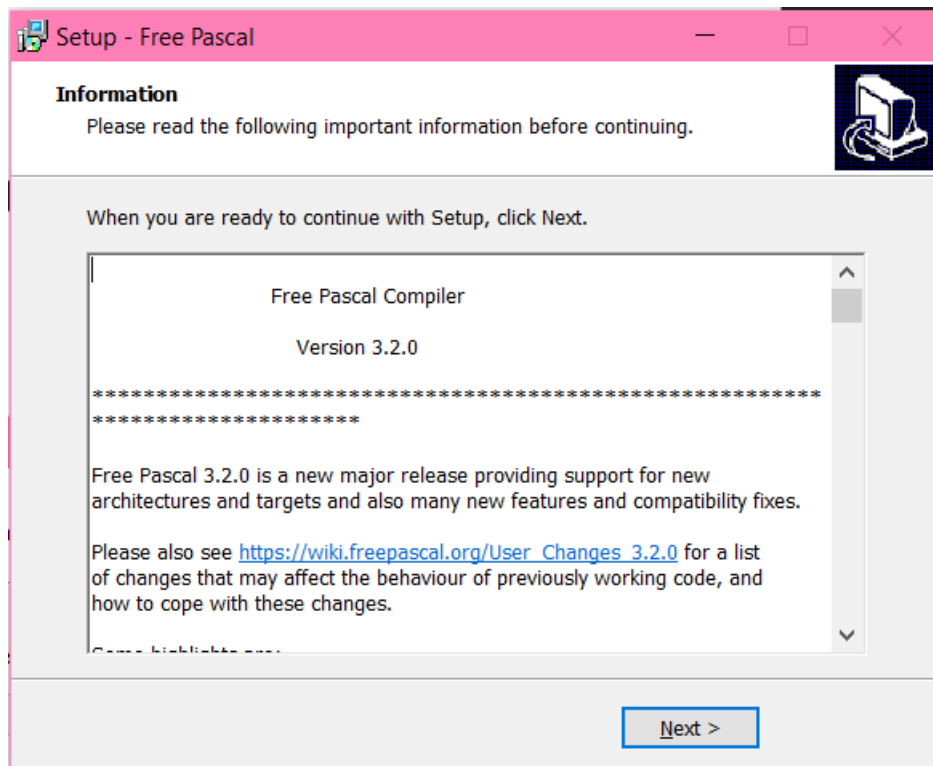
Kemudian lihat ringkasan instalasi, pastikan sudah benar sebelum proses instalasi dilanjutkan.



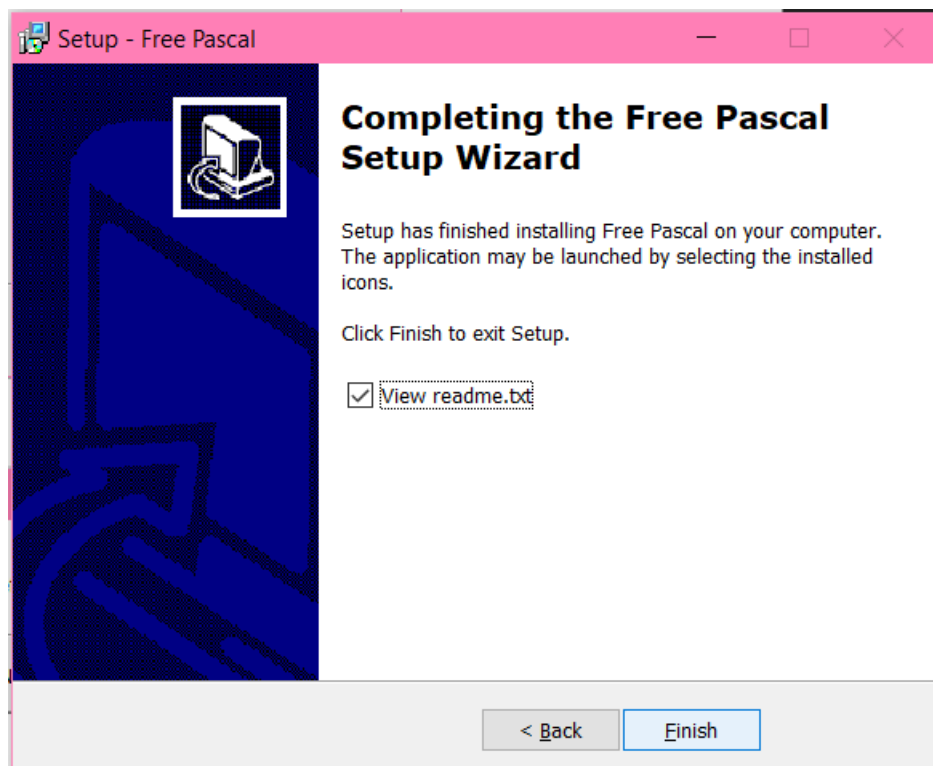
Jika sudah benar, klik install dan tunggu proses selesai.



Klik next.



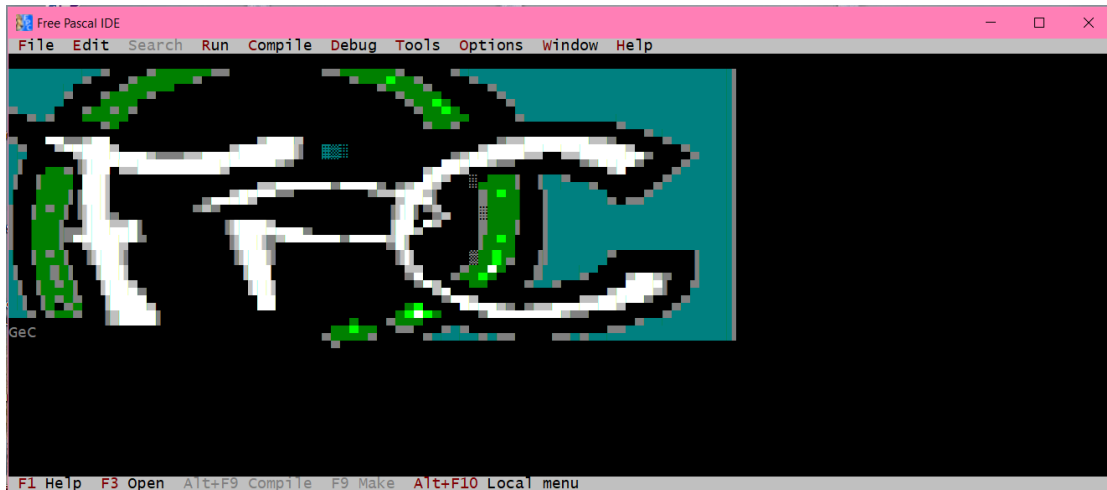
Setelah proses install selesai, klik finish.



Pergi ke desktop dan pastikan sudah ada shortcut Free Pascal IDE.



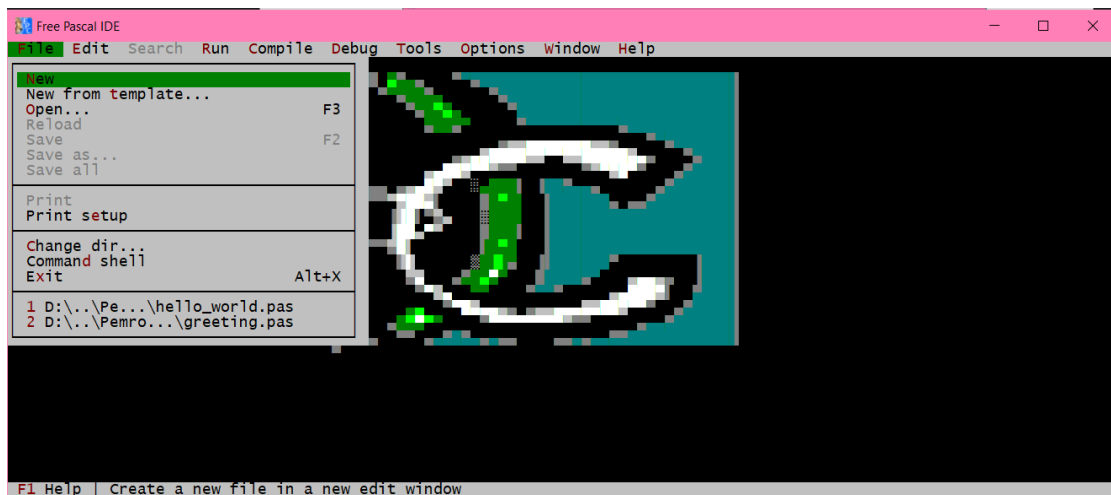
Untuk membuka Free Pascal, klik 2x pada icon. Dan akan muncul tampilan.



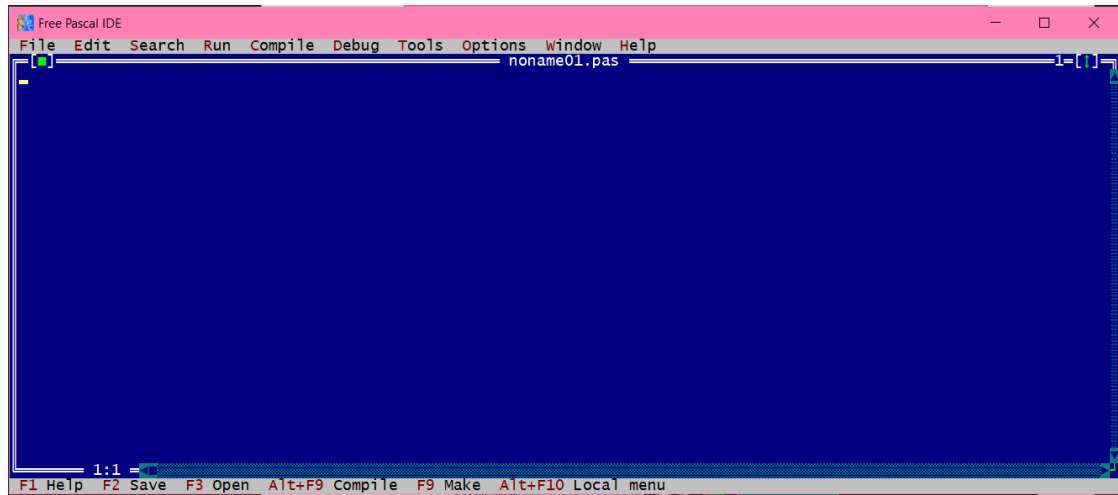
## ii) Membuat kode program

Dalam materi ini akan dijelaskan mengenai pengenalan Free Pascal.

1) Untuk membuat file baru, klik File >> New



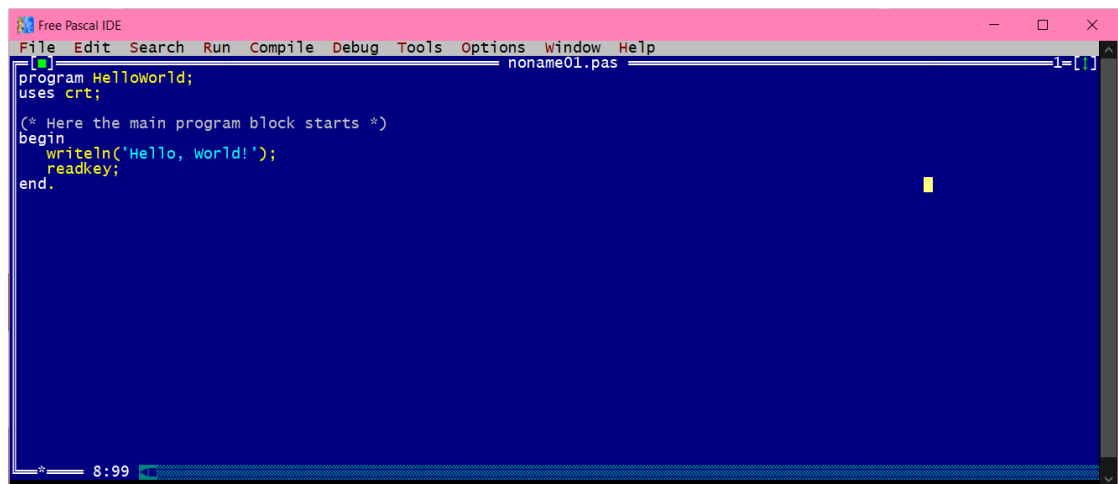
2) Akan muncul tampilan file baru tanpa nama (noname01.pas)



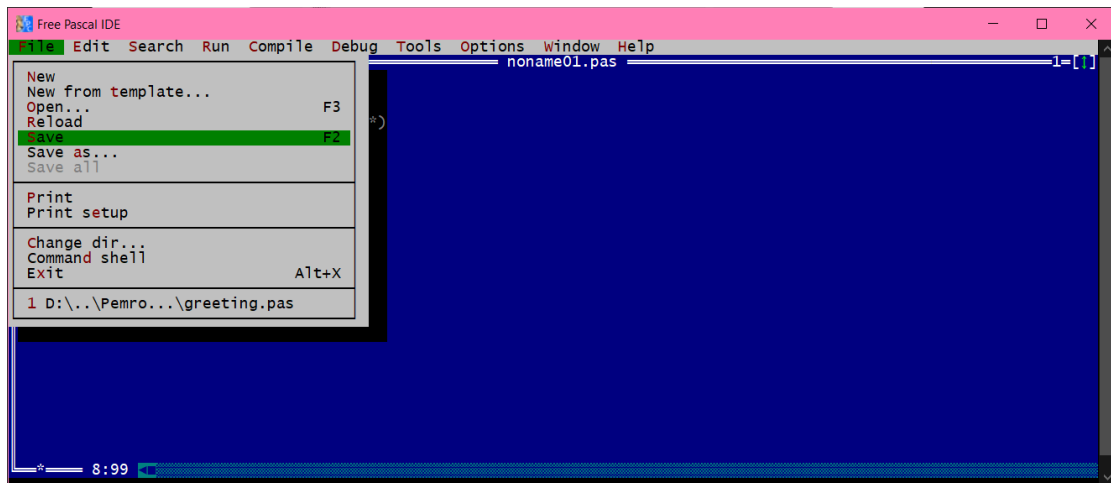
3) Ketikkan kode berikut

```
program HelloWorld;
uses crt;

(* Ini adalah awal blok program utama*)
begin
    writeln('Hello, World!');
    readkey;
end.
```

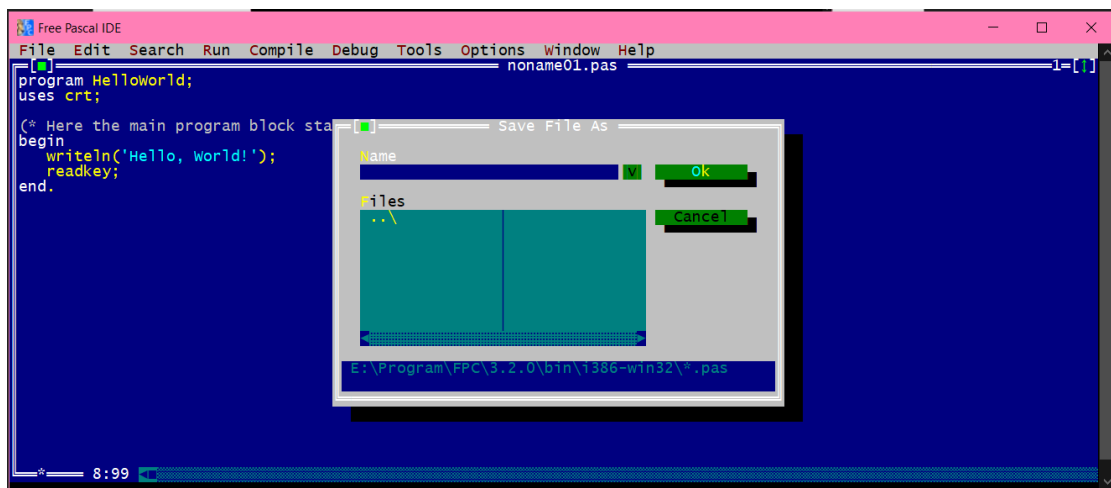


4) Untuk menyimpan file, Klik File >> Save atau tekan F2.

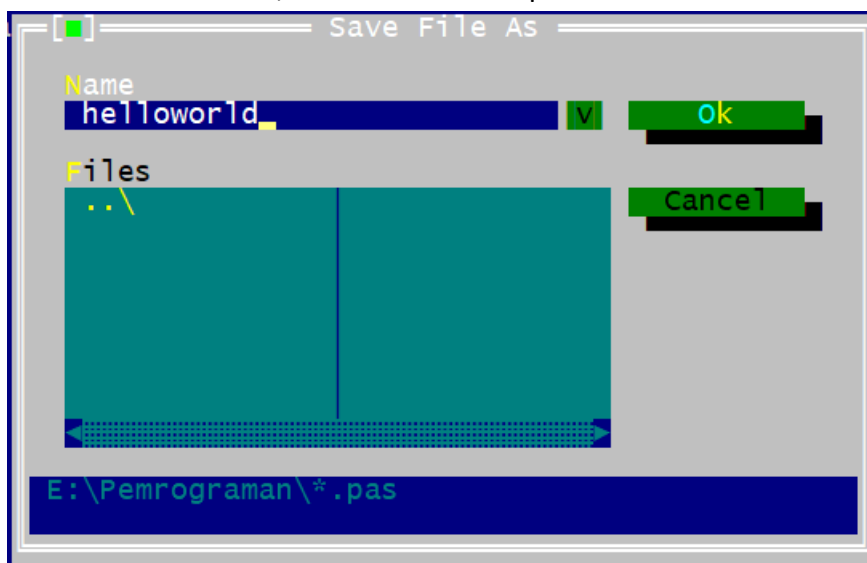


5) Lokasi default adalah %lokasiinstall%\FPC\3.2.0\bin\i386-win32.

Pilih lokasi folder, dengan tekan 2x pada ..\ untuk ke folder di level atas sampai ditemukan folder yang diinginkan pada drive yang sama dengan lokasi install.



Kemudian beri nama, misal "helloworld.pas". kemudian klik OK.

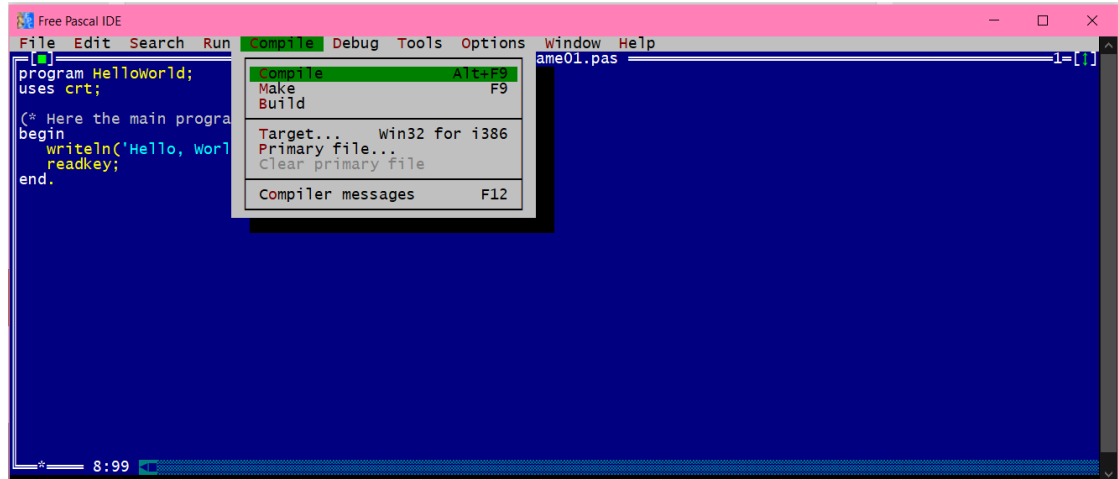


Jika berhasil maka di folder yang ditentukan akan ada 1 file "helloworld.pas"

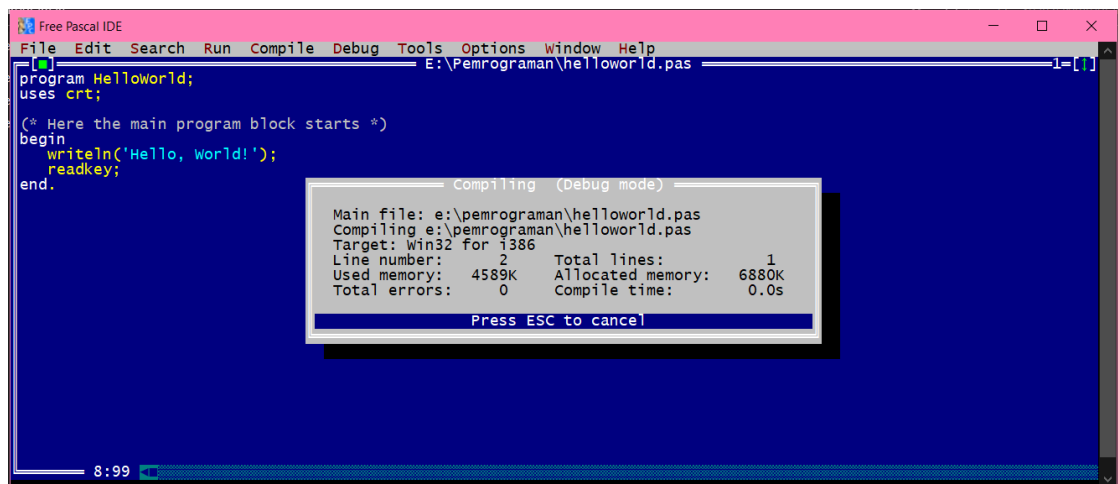


Name	Date modified	Type	Size
helloworld.pas	1/19/2021 11:20 AM	PAS File	1 KB

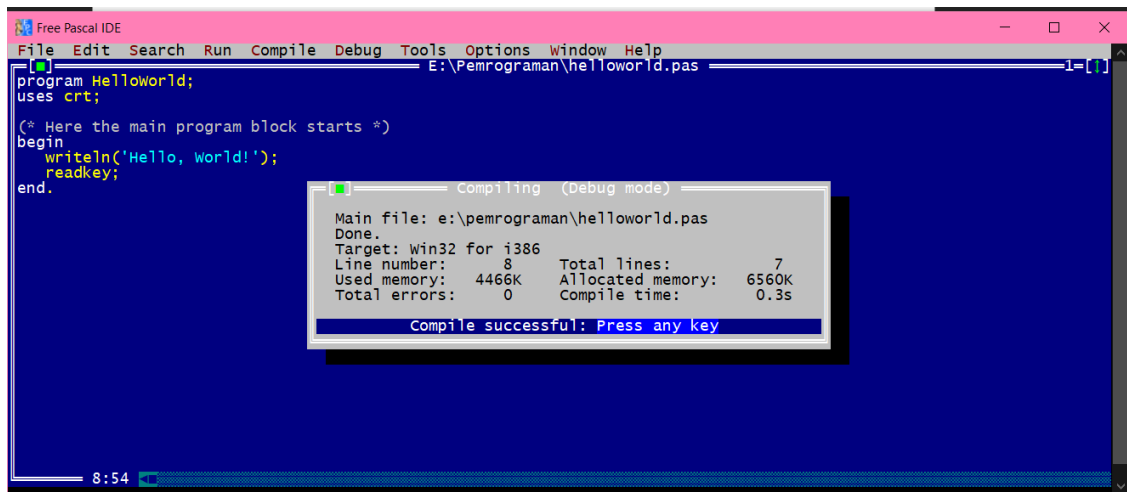
6) Untuk melakukan *compile*, Klik Compile >> Compile atau tekan ALT+F9.



7) Tunggu proses compile selesai.



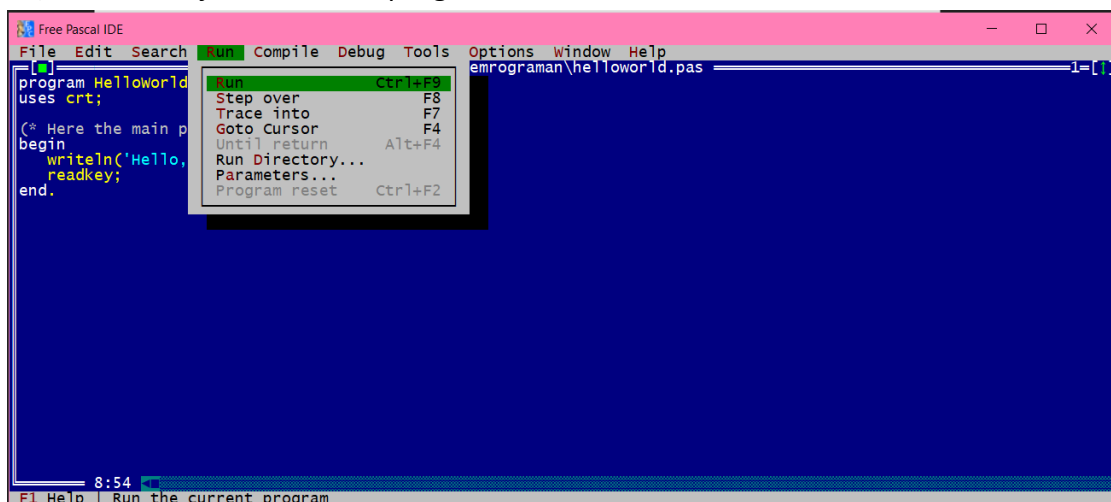
8) Jika berhasil maka tampilan menjadi seperti pada gambar, tekan sembarang tombol untuk kembali ke halaman kode. Namun jika gagal, maka akan ditampilkan daftar kesalahan yang ada.



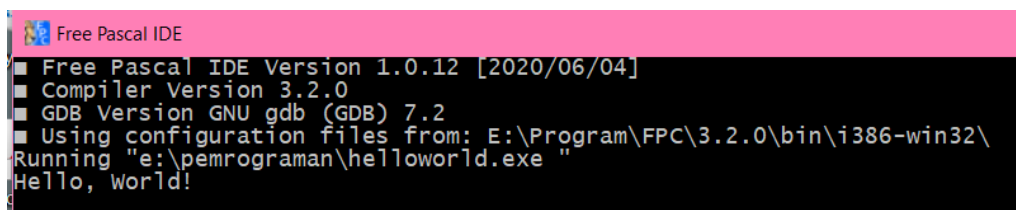
- 9) Jika berhasil maka di folder akan muncul 2 file tambahan dengan nama yang sama tetapi ekstensi \*.o dan \*.exe

Name	Date modified	Type	Size
helloworld.exe	1/19/2021 11:23 AM	Application	58 KB
helloworld.o	1/19/2021 11:23 AM	O File	4 KB
helloworld.pas	1/19/2021 11:20 AM	PAS File	1 KB

- 10) Untuk menjalankan kode program, Klik Run >> Run atau CTRL+F9



- 11) Tampilan hasilnya adalah seperti gambar.



Selain itu, terdapat pula layanan kompilasi dan interpretasi program Pascal secara online. Beberapa diantara dapat diakses dan digunakan melalui link berikut:

[https://www.onlinegdb.com/online\\_pascal\\_compiler](https://www.onlinegdb.com/online_pascal_compiler)

[https://rextester.com/l/pascal\\_online\\_compiler](https://rextester.com/l/pascal_online_compiler)

[https://www.tutorialspoint.com/compile\\_pascal\\_online.php](https://www.tutorialspoint.com/compile_pascal_online.php)

<https://www.jdoodle.com/execute-pascal-online/>

Pastikan untuk memperhatikan jenis dan versi compiler dan sintaks yang digunakan. Setiap compiler dan interpreter dapat memiliki karakteristik yang berbeda.

### Struktur Bahasa Pemrograman Pascal dan Perintah Standar Output

Secara ringkas, struktur program Pascal adalah sebagai berikut:

```
program {nama program}
uses {nama library yang digunakan, dipisahkan dengan koma}
const {deklarasi konstanta global}
var {deklarasi variabel global}

function {deklarasi fungsi, jika ada}
{ variabel lokal }
begin
...
end;

procedure {deklarasi prosedur, jika ada}
{ variabel lokal }
begin
...
end;

begin { awal blok kode utama program }
...
end. { akhir blok kode utama program }
```

Buat dan jalankan sintaks berikut pada compiler dan interpreter bahasa pemrograman Pascal:

```
Program tampil1;
Uses crt;
Begin
    Clrscr;
    Write ('Politeknik ');
    Write ('Statistika ');
Write ('STIS');
```

End.

Kemudian jalankan sintaks program berikut:

```
Program tampil2;  
Uses crt;  
Begin  
  Clrscr;  
  Writeln ('Politeknik');  
  Writeln ('Statistika');  
Writeln ('STIS');  
End.
```

Perintah/statement `Write` dan `Writeln` merupakan perintah standar output untuk mengatur tampilan dari program pada media output komputer, seperti monitor, printer, dan lain-lain. Pada percobaan ini, tampilan program ditampilkan pada layar monitor (console).

### Software untuk Pembuatan Flowchart

Tersedia berbagai jenis software untuk membantu pembuatan flowchart, diantaranya adalah:



Sebagai contoh, **yEd** dan **draw.io** masing-masing merupakan freeware (dapat digunakan tanpa membayar/membeli) yang tersedia dalam versi offline (diinstall pada PC/Laptop) maupun versi online (diakses dan digunakan dengan internet, tanpa diinstall). Software dan petunjuk instalasi dapat diakses melalui:

**yEd:** <https://www.yworks.com/products/yed>

**draw.io:** <https://www.diagrams.net/>

Cara penggunaan dapat dilihat pada link berikut:

yEd: <https://youtu.be/ujMhxPJnJCw>

draw.io: <https://youtu.be/Z0D96ZikMkc>

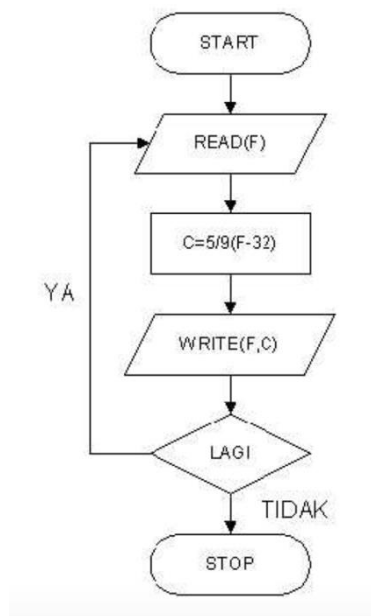
### Penugasan

1. Buatlah program dalam bahasa Pascal dengan output pada layar monitor berikut:

```
## ### ### ##
#   #   #   #
#   #   #   #
#   #   #   #
##   #   ### ##
```

```
42
--- x 52 + 6 x 3 + 4
7
```

2. Buatlah flowchart berikut ini dengan software yEd atau draw.io, kemudian export hasilnya ke dalam format file \*.png



## **MODUL 2: Pseudocode, Flowchart, Perintah Input, dan Operator Aritmatika**

---

### **2.1. Deskripsi Singkat**

Praktikum ini sebagai media bagi mahasiswa untuk mengetahui cara menjelaskan penyelesaian permasalahan berbasis algoritma dengan memanfaatkan tools berupa pseudocode dan flowchart. Selain itu, untuk lebih memperdalam kemampuan programming, pada modul praktikum ini juga diberikan latihan menggunakan perintah input sebagai kelanjutan perintah output pada modul sebelumnya. Operator aritmatika juga diperkenalkan pada praktikum ini.

### **2.2. Tujuan Praktikum**

Setelah praktikum pada modul 2 ini diharapkan:

1. Mahasiswa mampu menerjemahkan solusi penyelesaian kasus berbasis algoritma kedalam bentuk pseudocode dan flowchart
2. Mahasiswa memahami perintah input pada bahasa pemrograman Pascal
3. Mahasiswa memahami operator aritmatika pada bahasa pemrograman Pascal

### **2.3. Material Praktikum**

Tidak ada material khusus untuk praktikum ini.

### **2.4. Kegiatan Praktikum**

#### **A. Mengamati contoh flowchart dan pseudocode**

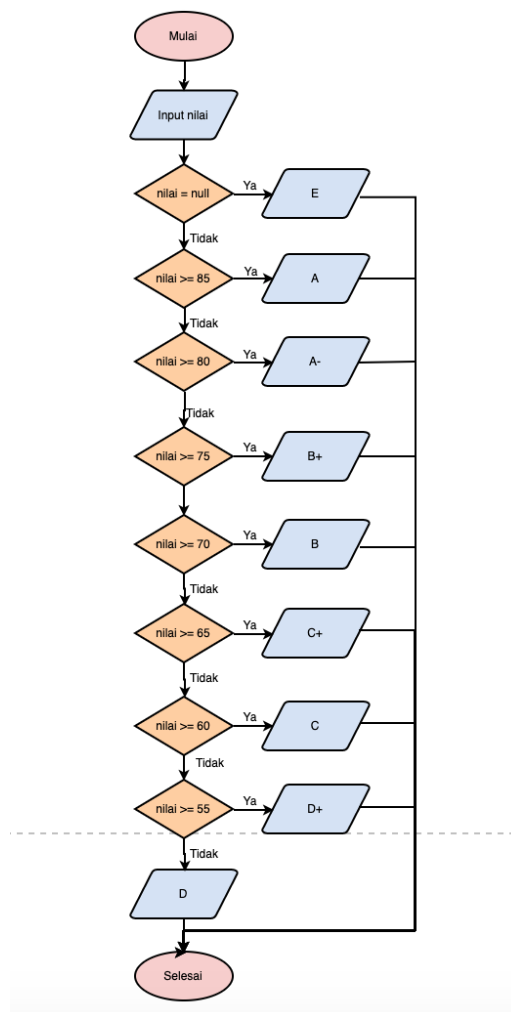
Proses, baik berupa kasus permasalahan ataupun solusinya, dapat dijelaskan secara algoritmik. Pseudocode dan flowchart dapat menjelaskan cara kerja suatu proses secara algoritmik. Sebagai contoh, program sederhana untuk konversi dari nilai absolut ke nilai mutu matakuliah dapat dijelaskan dengan pseudocode berikut:

```

integer nilai
Display "Inputkan nilai absolut: "
Input nilai
jika 85 <= nilai <= 100
    Display "Nilai mutu: A"
jika 80 <= nilai < 85
    Display "Nilai mutu: A-"
jika 75 <= nilai < 80
    Display "Nilai mutu: B+"
jika 70 <= nilai < 75
    Display "Nilai mutu: B"
jika 65 <= nilai < 70
    Display "Nilai mutu: C+"
jika 60 <= nilai < 65
    Display "Nilai mutu: C"
jika 55 <= nilai < 60
    Display "Nilai mutu: D+"
jika nilai < 55
    Display "Nilai mutu: D"
jika nilai = null //nilai tidak diinput
    Display "Nilai mutu: E"

```

serta dapat digambarkan dalam bentuk flowchart sebagai berikut:



Contoh lain flowchart dengan penggunaan beragam notasi untuk menggambarkan beragam proses yang algoritmik dapat dilihat pada link berikut:

<https://online.visual-paradigm.com/drive/#diagramlist:proj=0&new=Flowchart>

Pelajari penggunaan simbol dan notasi pada flowchart tersebut untuk memahami cara penggunaannya.

## **B. Perintah Input pada pemrograman Pascal**

Penggunaan perintah input pada program Pascal dapat dipelajari pada dokumentasi freepascal pada link berikut:

<https://wiki.freepascal.org/Input>

Buat, kompilasi, dan jalankanlah sintaks program Pascal berikut yang menerapkan perintah input:

```
Program input1;
Uses Crt;
Var
    nama,alamat: String;
    umur: Integer;
    ipk: Real;
Begin

    Clrscr;
    Writeln ('Masukkan Data Mahasiswa');
    Writeln ('=====');
    Write ('Nama :');
    Readln (nama);
    Write ('Alamat :');
    Readln (alamat);
    Write ('Umur   :');
    Read (umur);
    Write ('IPK :');
    Readln (ipk);
    Writeln;
```



```

Writeln ('=====HASIL=====');
Write ('Nama: ',nama,'. Alamat: ',alamat);
Writeln ('. Umur: ',umur,'. IPK: ',ipk:1:2);
Readln;
End.

```

Masukkan input melalui keyboard pada program yang telah dijalankan sesuai dengan tipe data yang dibutuhkan.

### C. Operator aritmatik

Penggunaan operator aritmatika pada program Pascal dapat dipelajari pada dokumentasi freepascal pada link berikut:

<https://www.freepascal.org/docs-html/ref/refse104.html>

Buat, kompilasi, dan jalankanlah sintaks program Pascal berikut yang menerapkan operator aritmatika input:

#### Contoh 2:

```

(*)
Diadopsi dari https://wiki.freepascal.org/Solution_2
*)

program hitungrerata;

const
    n = 5;

var
    A, B, C, D, E : integer;
    total : integer;
    rerata : real;

begin
    write ('Inputkan data pertama: ');
    readln (A);

```

```

write ('Inputkan data kedua: ');
readln (B);
write ('Inputkan data ketiga: ');
readln (C);
write ('Inputkan data keempat: ');
readln (D);
write ('Inputkan data kelima: ');
readln (E);
total := A + B + C + D + E;
rerata := total / 5;
writeln ('n = ', n);
writeln;
writeln ('Data pertama:', A:8);
writeln ('Data kedua:', B:8);
writeln ('Data ketiga:', C:8);
writeln ('Data keempat:', D:8);
writeln ('Data kelima:', E:8);
writeln ('=====');
writeln ('Total:', total:12);
writeln ('Rata-rata:', rerata:10:1);
end.

```

Masukkan input melalui keyboard pada program yang telah dijalankan sesuai dengan tipe data yang dibutuhkan.

## 2.5. Penugasan

1. Berikut adalah pseudocode untuk menghitung standard deviasi:

```

Step 1: Hitung nilai rata-rata.
Step 2: Untuk setiap item data, hitung selisihnya dengan
        rata-rata kemudian dikuadratkan.
Step 3: Hitung total nilai dari Step 2.
Step 4: Total nilai pada Step 3 dibagi dengan jumlah data.
Step 5: Hitung akar kuadrat dari Step 4.

```

Buatlah flowchart untuk pseudocode tersebut.

2. Buatlah program dengan bahasa pemrograman Pascal dengan perintah input dan tampilan berikut:

#### Program 1:

```
MENCARI KONVERSI SUHU
=====
Input derajat Celsius = -
Derjad Fahrenheit = -
```

Proses:

$$\text{Fahrenheit} = 9/5 * \text{Celsius} + 32$$

#### Program 2:

```
Menentukan Harga Bayar
=====
Jumlah barang = -
Harga satuan Rp. = -
Pajak Rp. = -
Harga Bayar Rp. = -
```

Proses:

$$\text{Harga bayar} = \text{Jumlah} \times \text{harga\_satuan} + \text{pajak}$$

#### Program 3:

```
Mencari Luas Lingkaran
=====
Input Panjang jari-jari (cm) = -
Luas adalah (cm2) = -
```

Proses:

$$\text{Luas} = 22/7 \times \text{jari} \times \text{jari}$$

## **2.6. Referensi**

[https://www.researchgate.net/publication/318722368 Buku Penunjang Praktek P  
ascal](https://www.researchgate.net/publication/318722368_Buku_Penunjang_Praktek_Pascal)

<https://online.visual-paradigm.com/>

<https://www.freepascal.org/>

## MODUL 3: Variabel ,Tipe Data, dan Fungsi Standar Matematika

---

### Deskripsi Singkat

Praktikum ini bertujuan memperkenalkan cara penggunaan variabel dan jenis-jenis tipe data yang didukung pada bahasa pemrograman Pascal sehingga mahasiswa mampu mengetahui karakteristik setiap tipe data serta menggunakannya dalam bentuk variabel. Pada praktikum ini, diperkenalkan juga beberapa fungsi standar matematika yang didukung bahasa pemrograman Pascal

### Tujuan Praktikum

Setelah praktikum pada modul 3 ini diharapkan:

1. Mahasiswa mampu mengetahui cara penggunaan variabel pada bahasa pemrograman Pascal
2. Mahasiswa memahami berbagai jenis tipe data yang didukung bahasa pemrograman Pascal
3. Mahasiswa mampu menggunakan fungsi standar matematika yang didukung bahasa pemrograman Pascal

### Material Praktikum

Tidak ada material khusus untuk praktikum ini.

### Kegiatan Praktikum

#### A. Mencoba berbagai jenis deklarasi variabel dan tipe data

Pada bahasa pemrograman Pascal, telah ditetapkan tipe data yang dapat digunakan beserta batasan nilainya yang dapat dipelajari pada link berikut:

[https://wiki.freepascal.org/Variables\\_and\\_Data\\_Types](https://wiki.freepascal.org/Variables_and_Data_Types)

Cobalah menulis program berikut. Lakukan kompilasi untuk setiap baris statement pemberian nilai variabel untuk mengetahui apakah berhasil atau tidak. Lengkapi sintaks dengan keterangan dalam bentuk komentar dengan menuliskan penyebabnya apabila pemberian nilai tidak berhasil.

```
PROGRAM Test1;  
VAR  
    x : REAL;  
    i : INTEGER;  
    c : CHAR;
```

```

    s : STRING;
BEGIN
    x := -34.55;
    x := -3.9E-3;
    WRITELN(x);
    i := 10;
    i := i * i;
    i := 9933;
    i := -99999;
    i := 999.44;
    c := '1';
    c := 1;
    c := 'Bert';
    c := 'd';
    WRITELN(c);
    d := 'c';
    WRITELN(s);
END.

```

### **Mencoba Perintah Input dan Format Output pada Variabel**

Perintah input dan output dapat digunakan untuk mengisi dan menampilkan nilai variabel serta melakukan format penampilan nilai variabel seperti yang dijelaskan pada link berikut:

[https://wiki.freepascal.org/Formatting\\_output](https://wiki.freepascal.org/Formatting_output)

Buat, kompilasi, dan jalankanlah sintaks program Pascal berikut kemudian amati outputnya untuk memahami maksud dari tiap statement:

```

PROGRAM Test2;
VAR
    x : REAL;
    i : INTEGER;
    j : INTEGER;
BEGIN
    READLN(x); { inputkan 12.449 pada keyboard }
    READLN(i); { inputkan 10 pada keyboard }
    READLN(j); { inputkan -300 pada keyboard }

```

```

WRITE('Format output');
WRITELN('Integer tidak terformat ',i);
WRITELN('Penghitungan integer tidak terformat ',i*i);
WRITELN('Integer terformat ',i:4);
WRITELN('Integer terformat ',j:4);
WRITELN('Bilangan real tidak terformat ',x);
WRITE('Bilangan real terformat ');
WRITE(x:8:2);
END.

```

Masukkan input melalui keyboard pada program yang telah dijalankan sesuai dengan keterangan pada komentar.

### Penugasan

1. Jalankan statement bahasa pemrograman Pascal, kemudian tuliskan output masing-masing statetement.

	Statement	Output
<b>Integers</b>	Writeln(1234:7); Writeln(1234:6); Writeln(1234:4); Writeln(1234:1); Writeln(1234:0);	
<b>Real numbers</b>	Writeln(12.35:10:4); Writeln(12.35:10:3); Writeln(12.35:8:1); Writeln(12.35:8:0); Writeln(-12.35:10:1); Writeln(12.35:9); Writeln(12.35:7); Writeln(12.35:0:2);	
<b>Strings</b>	Writeln('a':6); Writeln('abc':6); Writeln('abc':9);	

2. Buatlah program dengan bahasa pemrograman Pascal dengan menggunakan perintah input dan tipe data yang tepat sebagai variabel. Gunakan fungsi standard yang terdapat pada bahasa pemrograman Pascal pada link berikut:

[https://wiki.freepascal.org/Standard\\_Functions](https://wiki.freepascal.org/Standard_Functions)

untuk membantu pemrosesan data.

#### Program 1:

```
Mencari Standar Deviasi (SD)
=====
Data ke x (data) = -
Rata - rata (rata) = -
Jumlah Data (n) = -
Standar Deviasi (sd)= -
```

Proses:

$$sd = \sqrt{\frac{(data - rata)^2}{n - 1}}$$

#### Program 2:

```
NILAI TABEL DARI DISTRIBUSI NORMAL
~~~~~
Nilai rata-rata masing-masing (x) = -
Nilai rata-rata keseluruhan (m) = -
Data ke x (data)= -
Jumlah data (n) = -
Nilai Tabel (z) = -
```

Proses:

$$z = \frac{x - m}{\sqrt{\frac{(data - x)^2}{n - 1}}}$$

#### **Referensi**

1. [https://www.researchgate.net/publication/318722368\\_Buku\\_Penunjang\\_Praktek\\_Pascal](https://www.researchgate.net/publication/318722368_Buku_Penunjang_Praktek_Pascal)
2. <https://www.freepascal.org/>
3. <http://eddiejackson.net/>
4. <https://www.i-garden.org/>

## MODUL 4 STRUKTUR PEMILIHAN/PERCABANGAN IF-THEN

---

### Deskripsi Singkat

Pemilihan kondisi adalah proses penentuan langkah berikutnya berdasarkan proses yang terjadi sebelumnya. Pemilihan kondisi ini sangat penting dalam pemrograman sebab dengan adanya seleksi kondisi, program dapat menentukan proses apa yang harus dilakukan selanjutnya berdasarkan keadaan sebelumnya. Sehingga nampak seolah-olah program dapat berpikir dan mengambil keputusan. Disinilah letak kekurangan komputernya itu tidak mampu berpikir sendiri, semua hal yang dilakukan adalah berdasarkan perintah. Dalam memprogram seringkali digunakan suatu percabangan untuk pengambilan keputusan dari sejumlah pilihan yang mungkin.

### Tujuan Praktikum

Setelah praktikum pada modul 6 ini diharapkan mahasiswa mempunyai kompetensi untuk memahami dan menguasai kondisional pada Pascal, membuat suatu logika kondisional untuk menyelesaikan suatu permasalahan, dan membuat program dengan mengimplementasikan struktur pemilihan/percabangan if then.

### Material Praktikum

Kegiatan pada modul 4 ini memerlukan material berupa software editor program Pascal dan software perancangan diagram.

### Kegiatan Praktikum

Dalam materi ini akan dijelaskan mengenai struktur pemilihan/percabangan dalam pemrograman dengan if then. Perintah percabangan diperlukan akan dilakukan aksi ketika sebuah kondisi terpenuhi maka jalankan kode program. Jika kondisi tidak terpenuhi, jalankan kode program yang lain. Di dalam Pascal terdapat beberapa struktur kondisi if then yakni:

#### A. IF - THEN

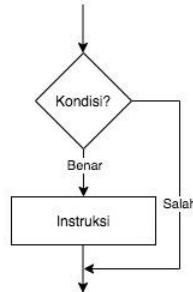
Konsep dasar dari percabangan perintah IF THEN dalam bahasa pemrograman Pascal adalah sebagai berikut:

```
IF (kondisi) THEN
begin
    (statement)
end;
```



Kondisi berperan sebagai penentu dari struktur percabangan ini. Jika kondisi terpenuhi (menghasilkan nilai TRUE), kode program akan dijalankan. Jika kondisi tidak terpenuhi (menghasilkan nilai FALSE), tidak terjadi apa-apa.

Bentuk flowchart dari bentuk IF kondisi THEN statement adalah sebagai berikut:



Contoh 1: Contoh program yang menggunakan bentuk umum percabangan IF kondisi THEN statement

```
program struktur_if_then;
uses crt;
var
    nilai: integer;
begin
    clrscr;
    nilai := 100;
    if (nilai > 50) then
        begin
            writeln('Nilai "UTS" lebih besar dari 50');
        end;
    writeln('Mahasiswa D-III Statistika Polstat STIS');
    readln;
end.
OUTPUT:
Nilai "UTS" lebih besar dari 50
Mahasiswa D-III Statistika Polstat STIS
```

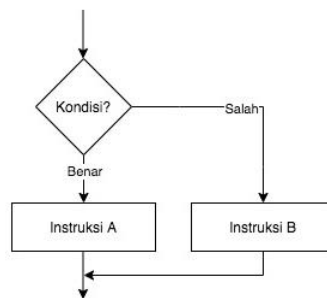
\* Saran: Selalu buat blok begin dan end; ketika membuat sebuah percabangan. Hal ini dilakukan untuk memastikan baris mana saja yang termasuk ke dalam blok if.

## B. IF-THEN-ELSE

Kondisi IF THEN ELSE adalah tambahan dari kondisi IF THEN. Bagian ELSE digunakan untuk menjalankan kode program apabila sebuah kondisi tidak terpenuhi. Konsep dasar dari percabangan IF THEN ELSE dalam bahasa pemrogram Pascal adalah sebagai berikut:

```
IF (kondisi) THEN
begin
    (statement 1)
end
ELSE
begin
    (statement 2)
end;
```

Bentuk flowchart dari bentuk IF kondisi THEN statement ELSE Statement 2 adalah sebagai berikut:



Contoh 2: Contoh program yang menggunakan bentuk umum percabangan IF kondisi THEN ELSE statement

```
program struktur_if_then_else;
uses crt;
var
    nilai: integer;
begin
    clrscr;
    nilai := 45;
    if (nilai > 50) then
    begin
        writeln('Nilai "UTS" lebih besar dari 50');
    end
```

```

else
    begin
        writeln('Nilai "UTS" lebih kecil dari 50');
    end;
    readln;
end.

OUTPUT:
Nilai "UTS" lebih kecil dari 50

```

Catatan: Penutup block sebelum perintah ELSE tidak boleh ada titik koma “;”

### C. IF Bersarang (Nested IF)

IF bersarang atau nested IF adalah penggunaan struktur IF di dalam IF. Kondisi seperti ini sering digunakan untuk kode program yang sudah cukup kompleks. Banyak variasi dari nested IF, tergantung kode program yang ingin dirancang. Salah satunya adalah sebagai berikut:

```

IF (kondisi 1) THEN
    begin
        (statement 1)
        IF (kondisi 1.1) THEN
            begin
                (statement 1.1)
            end;
        end
    ELSE
        begin
            (statement 2)
        end;

```

Keterangan: Sebuah kondisi IF di dalam IF (nested IF). Statement 1.1 hanya akan dijalankan jika kondisi 1 dan kondisi 1.1 terpenuhi. Jika kondisi 1 tidak terpenuhi, program akan langsung lompat ke bagian ELSE.

Variasi nested IF berikutnya adalah sebagai berikut:

```

IF (kondisi 1) THEN
  begin
    (statement 1)
    IF (kondisi 1.1) THEN
      begin
        (statement 1.1)
        IF (kondisi 1.1.1) THEN
          begin
            (statement 1.1.1)
          end;
        end;
      end;
    end
  ELSE
    begin
      (statement 2)
      IF (kondisi 2.1) THEN
        begin
          (statement 2.1)
        end;
      end;
    end;
  end;

```

Keterangan: Struktur IF di dalam IF di dalam IF (2 level nested IF). Untuk program di atas diperlukan kehati-hatian dalam menentukan statement mana yang masuk ke dalam blok IF.

Penulisan indenting (meletakkan statement dengan mendahului penambahan beberapa spasi di awal) juga akan membantu untuk membedakan statement ini masuk IF yang ini, dan statement itu masuk ke IF yang itu.

Contoh 3: contoh program nested IF

```

program struktur_if_then_else_nested;
uses crt;
var
  angka:integer;
begin
  clrscr;

```

```

write('Masukkan sebuah angka: '); readln(angka);
if (angka mod 2 = 0) then
begin
    write('Angka yang anda masukkan
    merupakan bilangan genap');
    if (angka > 10) then
    begin
        writeln('dan besar dari 10');
    end
    else
    begin
        writeln('dan kecil dari 10');
    end;
end
else
begin
    write('Angka yang anda masukkan
    merupakan bilangan ganjil');
    if (angka > 10) then
    begin
        writeln('dan besar dari 10');
    end
    else
    begin
        writeln('dan kecil dari 10');
    end;
end;
readln;
end.

```

OUTPUT 1:

Masukkan sebuah angka : 7

Angka yang anda masukkan merupakan bilangan ganjil dan kecil dari 10

OUTPUT 2:

Masukkan sebuah angka : 12

Angka yang anda masukkan merupakan bilangan genap dan besar dari 10

### Penugasan

1. Buatlah program untuk menentukan bonus pegawai, berdasarkan ketentuan yang diberikan oleh bagian personalia dan keuangan sebagai berikut : Pegawai perusahaan digolongkan menjadi dua golongan, yaitu staf dan non staf. Staf akan mendapatkan bonus sebesar 1 juta rupiah dengan syarat bahwa ia telah bekerja paling tidak 5 tahun dan umurnya sudah mencapai 50 tahun; staf yang bekerja kurang dari 5 tahun dan berapapun umurnya, hanya mendapat bonus sebesar Rp. 500.000. Pegawai non staf yang telah bekerja 5 tahun atau berumur 50 tahun atau lebih akan mendapat bonus sebesar Rp. 400.000, selain itu akan mendapat bonus Rp. 250.000.
2. Buatlah program menggunakan statemen if then yang membaca sebuah besaran integer yang menunjukkan hasil pengukuran suhu pada derajat celcius dan menuliskan kondisi air pada suhu yang diberikan dengan ketentuan sebagai berikut :
  - a. Suhu sama atau kurang dari 0 (nol) tuliskan 'beku'.
  - b. Suhu lebih dari 0 (nol) dan kurang dari 100 tuliskan 'cair', dan
  - c. Suhu sama atau lebih dari 100 tuliskan 'uap'.
3. Buatlah program menggunakan statemen if then yang menerima masukan nama hari (string) dan menuliskan mata kuliah yang diberikan pada hari tersebut, dengan ketentuan sebagai berikut :

Senin mata kuliah yang diberikan Algoritma dan Pemrograman I  
Selasa mata kuliah yang diberikan Kalkulus  
Rabu mata kuliah yang diberikan Bahasa Indonesia  
Kamis mata kuliah yang diberikan Pengantar Teknik Informatika  
Jum'at mata kuliah yang diberikan Bahasa Inggris  
Sabtu mata kuliah yang diberikan Basis Data

## MODUL 5 MULTIPLE SELECTION (STRUKTUR PILIHAN MAJEMUK)

---

### 5.1 Deskripsi Singkat

Dalam pembuatan program, kadang kala kita dihadapkan pada banyak pilihan yang dapat dilakukan untuk suatu keadaan. Tentu saja persoalan seperti ini dapat diselesaikan dengan menggunakan IF. Akan tetapi pada kondisi tertentu akan lebih mudah dan lebih singkat jika menggunakan statemen Case.

### 5.2. Tujuan Praktikum

Setelah praktikum pada modul 5 ini diharapkan mahasiswa mempunyai kompetensi memahami dan menggunakan Struktur Case dalam program Pascal.

### 5.3. Material Praktikum

Dalam materi ini akan dijelaskan mengenai struktur program dalam Pascal dan contoh programnya.

### 5.4. Kegiatan Praktikum

Dalam materi ini akan dijelaskan mengenai Struktur Case sederhana dan Struktur Case Bersarang dalam Bahasa Pascal.

#### A. Statemen Case

Struktur pilihan Case mirip dengan Struktur pilihan IF yang berulang. Jika dalam Struktur IF penulisannya seperti berikut:

```
IF (kondisi1)
THEN
    (statemen1)
ELSE IF
(kondisi2) THEN
    (statemen2)
ELSE IF (kondisi3) THEN
    (statemen3);
```

Maka dengan Struktur CASE format penulisannya menjadi seperti berikut:

```
Case (ekspresi)
OF
    hasil1: statemen1;
    hasil2: statemen2;
    hasil3: statemen3;
End;
```

Atau

```
IF (kondisi1)
THEN

(statement1)
ELSE IF
(kondisi2) THEN
    (statement2)
ELSE IF (kondisi3) THEN
    (statement3)
ELSE (statement4);
```

Maka dengan Struktur CASE format penulisannya menjadi seperti berikut:

```
Case (ekspresi)
OF   hasil1: statemen1;
     hasil2: statemen2;
     hasil3: statemen3;
     ELSE statemen4;
End;
```

**Catatan:** ekspresi adalah nilai dari variabel atau hasil operasi suatu variabel dengan operator-operator yang bertipe ordinal yang akan diperiksa nilainya. Jika nilai ekspresi ini sama dengan hasil1 maka yang akan dijalankan adalah statemen1. Kondisi dapat ditulis dalam nilai tunggal atau sub jangkauan. kondisi1 yg ada pada IFTHEN sama dengan ekspresi=hasil1 pada CASE, sehingga IFTHEN dapat ditulis lebih detil menjadi:

```
IF (ekspresi=hasil1)
THEN

(statement1)
ELSE IF
(ekspresi=hasil2) THEN
    (statement2)
ELSE IF (ekspresi=hasil3) THEN
    (statement3)
ELSE (statement4);
```

Contoh tanpa ELSE:

```
{Program Konversi Nilai CASE}

{ 1}Program Contoh51;
{ 2}var  Nilai : integer;
```



```

{ 3}      grade : char;
{ 4}begin
{ 5}      writeln('Input nilai yang Anda dapatkan 0 s.d. 100
');
{ 6}      write('Nilai Anda = ');
{ 7}      readln(Nilai);
{ 8}
{ 9}      case Nilai of
{10}          0..59:  grade:='D';
{11}          60..69:  grade:='C';
{12}          70..79:  grade:='B';
{13}          80..100: grade:='A';
{14}      end;
{15}      if (nilai>=0) and (nilai <=100) then
{16}          writeln('Grade Anda', Grade )
{17}      else writeln('Anda salah input nilai');
{18}
{19}      Case Grade of
{20}          'D': writeln(' Kurang memuaskan');
{21}          'C': writeln(' Cukup');
{22}          'B': writeln(' Baik');
{23}          'A': writeln(' Sangat Baik');
{24}      end;
{25}      readln
{26}end.

```

Kompilasi dan jika tidak ada eror maka eksekusi program Contoh51 di atas. Ujicoba dengan segala kemungkinan nilai termasuk yang di luar 0 s.d 100. Bagaimana hasilnya untuk nilai dari 0 s.d 100? bagaimana juga hasilnya untuk nilai diluar 0 s.d. 100?

Contoh dengan ELSE

```

{Program Konversi Nilai CASE-ELSE}
{ 1}Program Contoh52;
{ 2}var  Nilai : integer;
{ 3}      grade : char;
{ 4}begin
{ 5}      writeln('Input nilai yang Anda dapatkan 0 s.d. 100
');
```

```

{ 6}  write('Nilai Anda = ');
{ 7}  readln(Nilai);
{ 8}
{ 9}  case Nilai of
{10}      0..59:  grade:='D';
{11}      60..69:  grade:='C';
{12}      70..79:  grade:='B';
{13}      80..100: grade:='A';
{14}      else grade:='F';
{15}  end;
{16}  if Grade='F' then
{17}      writeln('Anda salah input nilai')
{18}  else if writeln('Grade Anda', Grade );
{19}
{20}  Case Grade of
{21}      'D': writeln(' Kurang memuaskan');
{22}      'C': writeln(' Cukup');
{23}      'B': writeln(' Baik');
{24}      'A': writeln(' Sangat Baik');
{25}  end;
{26}  readln
{27}end.

```

Lakukan hal yang sama seperti program Contoh51 untuk program di atas. Tuliskan perbedaannya dalam catatan anda.

## B. Case bersarang (Nested Case)

Dalam pembuatan program, saat dihadapkan pada pilihan yang banyak, untuk mempermudah kadangkala dapat dibuat menjadi berjenjang sehingga di dalam CASE ada CASE lagi.

### Contoh:

```

{Program Menentukan Hasil Ujian}
{ 1}Program Contoh53;
{ 2}var
{ 3}    Nilai : integer;
{ 4}

```

```

{ 5}begin
{ 6}   writeln('Input nilai yang Anda dapatkan 0 s.d. 100
');
{ 7}   write('Nilai Anda = ');
{ 8}   readln(Nilai);
{ 9}
{10}   case nilai of
{11}       0..59   : writeln(' Anda tidak lulus';
{12}       60..100 :
{13}       begin
{14}           write('Anda lulus ');
{15}           case nilai of
{16}               60..74: writeln('dengan nilai cukup');
{17}               75..89: writeln('dengan nilai baik');
{18}               else writeln('dengan nilai sangat baik');
{19}           end;
{20}       end;
{21}       else writeln(' Anda salah input nilai');
{22}   end;
{23}
{24}   readln;
{25}end.

```

Eksekusi program Contoh53 setelah tidak ada error. Kemudian inputkan nilai untuk semua kemungkinan ada. Perhatikan seperti apa outputnya.

### Penugasan

- 1) Periksa ekspresi yang ada pada Struktur CASE, apakah boleh bertipe selain ordinal dengan beberapa compiler Pascal yang berbeda?
- 2) Apabila `Nilai` bertipe real, bagaimana caranya agar dapat menggunakan Struktur CASE.
- 3) Ubah Struktur CASE program di atas (ada 3 program) dengan Struktur pilihan yang lain.
- 4) Perbaiki ketiga contoh program di atas untuk bagian menginput nilai, baris 5-7 untuk contoh51 dan contoh52 serta baris 6-8 untuk contoh53 sehingga jika pengguna menginputkan nilai diluar 0 s.d. 100 maka program langsung memberikan pesan salah menginput nilai kemudian selesai.

- 5) Modifikasi Contoh53 sehingga nested(Bersarang) CASE berubah menjadi single CASE tanpa mengubah output program.
- 6) Buat catatan kelebihan dan kekurangan Struktur CASE dibandingkan dengan Struktur pilihan yang lain.

## MODUL 6 PENGULANGAN 1

---

### 6.1 Deskripsi Singkat

Dalam modul ini, akan dipelajari Struktur pengulangan sederhana dengan menggunakan FOR, WHILE-DO dan REPEAT-UNTIL

### 6.2 Tujuan Praktikum

Setelah praktikum pada modul 6 ini diharapkan mahasiswa mempunyai kompetensi yaitu memahami Struktur pengulangan FOR, WHILE-DO dan REPEAT-UNTIL dan dapat menggunakannya dalam program.

### 6.3 Material Praktikum

Kegiatan pada modul 6 ini memerlukan material berupa penjelasan singkat mengenai Struktur pengulangan dan contoh program dalam Pascal.

### 6.4 Kegiatan Praktikum

#### A. Struktur FOR

Secara garis besar, Struktur FOR terbagi ke dalam dua bentuk, Struktur FOR menaik dan Struktur For menurun. Dalam banyak keadaan, keduanya dapat digunakan pada kasus yang sama walaupun pada kondisi-kondisi tertentu lebih mudah atau lebih tepat menggunakan salah satunya.

Format FOR Menaik: (FOR-TO)

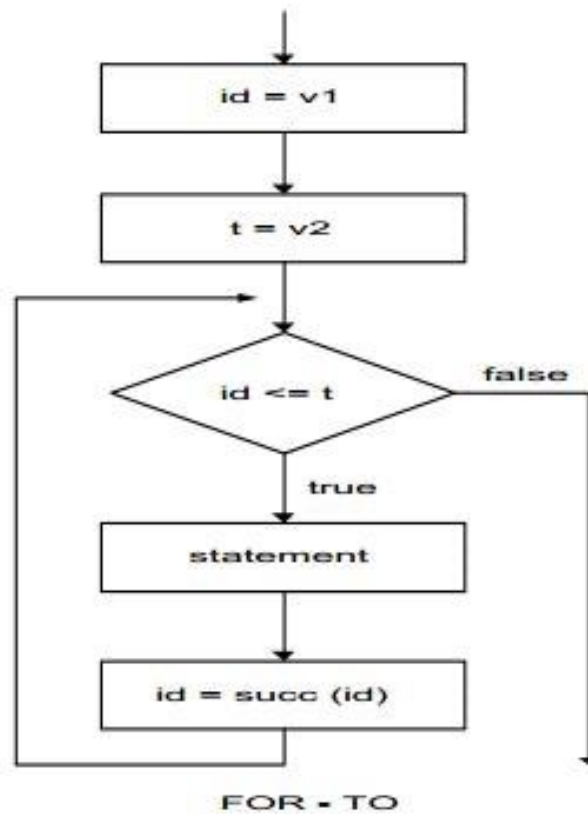
Bentuk umum *pseudocode*:

```
for id ← nilaiawal to nilaiakhir do  
    aksi  
endfor
```

Translasi *pseudocode* pada bahasa pascal:

```
for id:=v1 to v2 do  
    statement;
```

atau dapat digambarkan dengan flowchart seperti di bawah ini.



Format FOR Menurun: (FOR-DOWNT0)

Bentuk umum *pseudocode*:

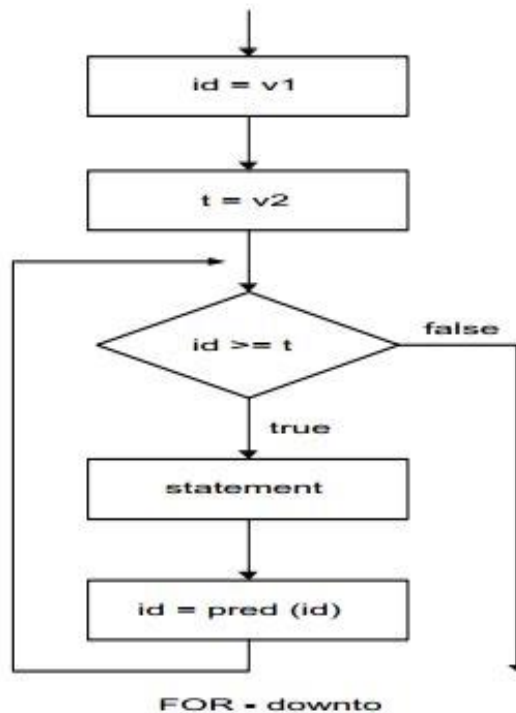
```

for id ← nilaiawal downto nilaiakhir do
  aksi
endfor
  
```

Translasi *pseudocode* pada bahasa pascal:

```

for id:=v1 DOWNto v2 do
  statement;
endfor
  
```



Catatan:

1. Variabel id harus bertipe ordinal.
2. Perbedaan antara FOR-TO dan FOR-DOWNTO dapat dilihat pada flowchart yang diberi tanda garis dan lingkaran merah.
3. Untuk FOR-TO, statement akan dieksekusi secara berulang selama  $id \leq t$  dan jika  $v1 > v2$  maka otomatis statement tidak dieksekusi sekalipun. Berlaku kebalikannya untuk FOR-DOWNTO.

Contoh pseudocode FOR-TO:

```

algoritma cetak_halo;
{mencetak 'HALO' sebanyak 10 kali}
deklarasi
  n:integer {pencacah pengulangan}
deskripsi
  for n ← 1 to 10 do
    write('HALO')
  endfor
{kondisi berhenti: n>10}
  
```

terjemahan dalam bahasa Pascal:

```
{Program Cetak Halo FOR menaik}
{ 1}program Contoh61;
{ 2}var n:integer;
{ 3}begin
{ 4}    for n:=1 to 10 do
{ 5}        writeln('HALO');
{ 6}    readln;
{ 7}end.
```

Contoh FOR-DOWNTO:

Pseudocode:

```
algoritma cetak_halo;
{mencetak 'HALO' sebanyak 10 kali}
deklarasi
    n:integer {pencacah pengulangan}
deskripsi
    for n ← 10 downto 1 do
        write('HALO')
    endfor
{kondisi berhenti: n<1}
```

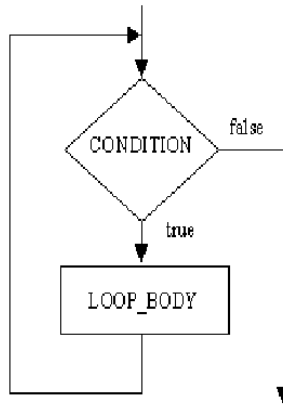
terjemahan dalam bahasa Pascal:

```
{Program Cetak Halo For Menurun}
{ 1}Program Contoh62;
{ 2}var n:integer;
{ 3}begin
{ 4}    for n:=10 downto 1 do
{ 5}        writeln('HALO');
{ 6}    readln;
{ 7}end.
```

## B. Struktur WHILE-DO

Bentuk flowchart:





Bentuk umum *pseudocode*:

```

while <kondisi> do
  aksi
endwhile

```

Translasi *pseudocode* pada bahasa pascal:

```

while kondisi do
  aksi;

```

Ketika instruksi dijalankan, pertama akan dicek pada bagian kondisi. Jika kondisi bernilai **true** maka aksi akan dijalankan. Aksi akan terus dijalankan berulang selama kondisi bernilai true. Jika kondisi sudah bernilai false maka aksi tidak akan dijalankan lagi. Dengan kata lain, while kondisi=true do aksi. Perhatikan algoritma cetak halo dan translasinya dalam bahasa pascal berikut ini:

```

algoritma cetak_halo;
{mencetak 'HALO' sebanyak 10 kali}
deklarasi
  n:integer {pencacah pengulangan}
deskripsi
  n ← 1
  while n<=10 do
    write('HALO')
  n ← n+1;
  endwhile
  {kondisi berhenti: n>10}

```

### Translasi algoritma Cetak Halo ke Bahasa Pascal:

```
{Program Cetak Halo WHILE-Do}
{ 1}program Contoh6.3;
{ 2}var n:integer;
{ 3}begin
{ 4}    n:=1;
{ 5}    while n<=10 do
{ 6}    begin
{ 7}        writeln('HALO')
{ 8}        n:=n+1;
{ 9}    end;
{10}    readln;
{11}end.
```

#### Penjelasan Program Cetak Halo:

Perhatikan bahwa jika terdapat lebih dari satu statemen dalam badan pengulangan maka perlu diselimuti dengan `begin` dan `end`. Kemudian perhatikan antara bagian `begin` dan `end` di program utama:

- 1) Pada baris ke-4, `n:=1` artinya Pada mulanya `n` diisi nilai 1
- 2) Sebelum memasuki badan pengulangan, kondisi `n<=10` diperiksa apakah bernilai `true`
- 3) Karena `1<=10` bernilai `true` maka pernyataan `writeln('HALO')` dilaksanakan sehingga output yang tercetak:  
  
HALO
- 4) Selanjutnya pernyataan `n:=n+1` menyebabkan nilai `n` bertambah 1 menjadi 2, lalu siklus pengulangan dimasuki lagi.
- 5) Sebelum melaksanakan `writeln('HALO')`, kondisi pengulangan `n<=10` diperiksa terlebih dahulu.
- 6) Karena `2<=10` bernilai `true` maka badan pengulangan dimasuki, `writeln('HALO')` dilaksanakan sehingga output yang tercetak:  
  
HALO  
  
HALO
- 7) Demikian seterusnya sebanyak 10 kali. Setiap kali badan pengulangan dimasuki, nilai `n` bertambah satu sampai `n=11`.

8) Karena  $11 > 10$  maka kondisi pengulangan  $n \leq 10$  bernilai `false`

9) Pengulangan dihentikan.

10) Jadi output program Cetak Halo di atas adalah adalah:

HALO

HALO

HALO

HALO

HALO

HALO

HALO

HALO

HALO

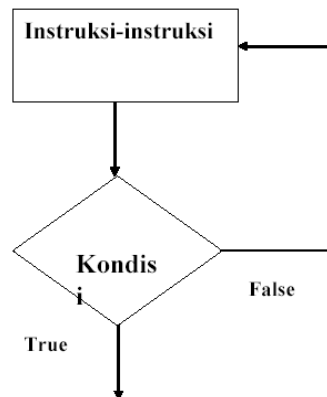
HALO

**Catatan:**

1. Apa yang terjadi Jika programmer lupa melakukan inisiasi nilai `n`?
2. Apa yang terjadi jika programmer lupa menambahkan nilai dengan satu setelah mencetak `Halo` ? Silahkan dicoba. Tapi jangan lupa **simpan** dulu programnya sebelum dijalankan.

### C. Struktur REPEAT-UNTIL

Bentuk flowchart:



Bentuk umum algoritma:

```
repeat
aksi
until <kondisi>
```

Translasi dalam Bahasa Pascal:

```
Repeat
aksi;
until kondisi;
```

- 1) Aksi di dalam badan pengulangan terus diulang sampai kondisi bernilai true
- 2) Dengan kata lain, jika kondisi berhenti masih false, pengulangan masih terus dilakukan. Repeat aksi until kondisi=true
- 3) Karena pengulangan harus berhenti, di dalam badan pengulangan harus ada aksi yang mengubah nilai kondisi

Algoritma cetak halo

```
algoritma cetak_halo;
{mencetak 'HALO' sebanyak 10 kali}
deklarasi
    n:integer {pencacah pengulangan}
deskripsi
    n ← 1
```

```

repeat
  write('HALO')
  n ← n+1;
until n > 10
{kondisi berhenti: n>10}

```

Translasi dalam bahasa pascal:

```

{Program Cetak Halo REPEAT-UNTIL}
{ 1}Program Contoh64;
{ 2}var n:integer;
{ 3}begin
{ 4}    n:=1;
{ 5}    repeat
{ 6}        writeln('HALO');
{ 7}        n:=n+1;
{ 8}    until n>10;
{ 9}    readln;
{10}end.

```

Struktur REPEAT-UNTIL memiliki makna yang mirip dengan WHILE-DO. Namun pada struktur REPEAT-UNTIL, aksi (atau sekumpulan aksi) dilaksanakan minimal satu kali, karena kondisi pengulangan diperiksa pada akhir struktur. Pada struktur WHILE-DO kondisi pengulangan diperiksa pada awal struktur sehingga memungkinkan badan pengulangan sama sekali tidak dilaksanakan.

#### D. Pemilihan Struktur yang tepat

Pada dasarnya ketiga struktur pengulangan ini dapat digunakan untuk kondisi yang sama, jadi tidak perlu bingung. Akan tetapi tentu saja ada kondisi-kondisi tertentu dimana lebih tepat kalau digunakan Struktur FOR misalnya, atau yang lain.

Struktur WHILE-DO dan REPEAT-UNTIL memiliki makna yang hampir sama, namun pemilihan struktur yang tepat bergantung pada masalah yang akan diprogram. Ingat bahwa pemeriksaan kondisi pada WHILE-DO dilakukan di awal pengulangan, Sedangkan pada REPEAT-UNTIL pemeriksaan kondisi dilakukan pada akhir pengulangan. Sebagai konsekuensi dari waktu pemeriksaan kondisi, aksi di dalam badan WHILE-DO paling sedikit dikerjakan 0 kali. Aksi di dalam badan REPEAT-UNTIL

paling sedikit dikerjakan 1 kali. Sedangkan FOR tepat digunakan untuk pengulangan yang sudah jelas berapa banyak akan dilakukan pengulangan.

## 6.5 Penugasan

- 1) Menggunakan while-do Buatlah program yang meminta masukan angka  $N$ . Program akan menghitung jumlah deret angka dari 1 sampai  $N$  dan menampilkan hasilnya. Misal  $N=5$ , maka  $1+2+3+4+5=15$ .
- 2) Menggunakan struktur while-do Buatlah program yang meminta masukan angka  $N$ . Program akan menuliskan teks lagu Anak Ayam Turun  $N$ . Misal jika  $N=5$  maka pada layar ouput:

```
Masukkan jumlah anak ayam: 5
Anak Ayam Turun 5
Anak Ayam turun 5, mati satu tinggal 4
Anak Ayam turun 4, mati satu tinggal 3
Anak Ayam turun 3, mati satu tinggal 2
Anak Ayam turun 2, mati satu tinggal 1
Anak Ayam turun 1, mati satu tinggal induknya.
```

- 3) Menggunakan struktur REPEAT-UNTIL, Buatlah program yang meminta masukan  $N$  dan masukan angka-angka sebanyak  $N$ . Program menghitung rata-rata dari  $N$  bilangan bulat yang dimasukkan oleh pengguna. Misal pengguna ingin menghitung rata-rata dari 3 bilangan. Maka kira-kira outputnya sebagai berikut:

```
Masukkan jumlah bilangan N: 3
Masukkan bilangan ke-1: 12
Masukkan bilangan ke-2: 14
Masukkan bilangan ke-3: 16
Rata-rata bilangan = 14
```

- 4) Buatlah program untuk menghitung jumlah deret:

$$1 - 1/3 + 1/5 - 1/7 + 1/9 - \dots \pm 1/N$$

Dimana  $N$  adalah bilangan ganjil positif yang dibaca dari masukan pengguna

- 5) Buatlah program yang meminta masukan  $N$  dan masukan angka-angka sebanyak  $N$ . Program akan menghitung jumlah nilai bilangan yang genap-

genap saja. Misal pengguna ingin memasukkan 3 bilangan. Maka kira-kira outputnya sebagai berikut:

```
Masukkan jumlah bilangan N: 3
Masukkan bilangan ke-1: 12
Masukkan bilangan ke-2: 14
Masukkan bilangan ke-3: 17
Jumlah bilangan genap = 26
```

- 6) Buatlah tabel harga fotokopian dari 1 –100 lembar, dimana harga perlembar kertas adalah 125 rupiah!

TABEL HARGA FOTOKOPI

LEMBAR	HARGA
1	125
2	250
3	325
.	.
.	.
.	.
100	12500

- 7) Dibaca data usia  $N$  orang mahasiswa STIS dari masukan. Buatlah program untuk menghitung jumlah mahasiswa yang berusia 17 tahun, 18 tahun, 19 tahun, dan 20 tahun, dan jumlah mahasiswa yang berusia selain itu. Sehingga kira-kira outputnya:

```
Masukkan jumlah mahasiswa : 3
Masukkan usia mahasiswa ke-1: 17
Masukkan usia mahasiswa ke-2: 19
Masukkan usia mahasiswa ke-3: 21
Jumlah mahasiswa berusia 17 tahun = 1
Jumlah mahasiswa berusia 18 tahun = 0
Jumlah mahasiswa berusia 19 tahun = 1
Jumlah mahasiswa berusia 20 tahun = 0
```

Jumlah mahasiswa berusia selain itu = 1

## MODUL 7 PENGULANGAN 2

---

### Deskripsi Singkat

Dalam modul 7 ini akan dipelajari statemen BREAK dan CONTINUE dalam struktur pengulangan serta Struktur Pengulangan Bersarang.

### Tujuan Praktikum

Setelah praktikum pada modul 7 ini diharapkan mahasiswa memahami dan mampu mengimplementasikan statemen BREAK dan CONTINUE serta Struktur Pengulangan Bersarang dalam program Pascal.

### Material Praktikum

Kegiatan pada modul 7 ini memerlukan material berupa penjelasan singkat serta beberapa contoh program Pascal.

### Kegiatan Praktikum

#### A. Statemen BREAK

Sesuai dengan namanya, perintah BREAK digunakan untuk berhenti. Berhenti dari apa? Berhenti dari perulangan yang sedang berlangsung. Apakah itu perulangan FOR, WHILE-DO maupun REPEAT-UNTIL, akan dipaksa berhenti saat ketemu perintah BREAK.

Umumnya perintah **BREAK** disimpan dalam sebuah kondisi IF-ELSE. Selama kondisi belum terpenuhi, perulangan tetap terus berlangsung. Jika kondisi sesuai, maka BREAK.

#### CONTOH :

```
{Contoh Program Break}

Program Contoh71;

var i:integer;

begin
    for i:=1 to 6 do
    begin
        if (i=4) then break;
        writeln('Coba Break ',i);
    end;
```



```
readln  
end.
```

**Hasil:**

```
Coba Break 1  
Coba Break 2  
Coba Break 3
```

## B. Statemen CONTINUE

Sesuai dengan namanya, perintah CONTINUE digunakan untuk melanjutkan ke perulangan berikutnya tanpa mengeksekusi statemen di bawahnya atau dengan kata lain hanya menghentikan satu kali iterasi saja saat perintah CONTINUE dijalankan.

Sama seperti BREAK, Umumnya perintah **CONTINUE** disimpan dalam sebuah kondisi IF-ELSE.

**CONTOH:**

```
{Contoh Program Continue}  
  
Program Contoh72;  
  
var i:integer;  
  
begin  
    for i:=1 to 6 do  
        begin  
            if (i=4) then continue;  
            writeln('Coba continue ',i);  
        end;  
        readln  
    end.  
end.
```

**Hasil:**

```
Coba continue 1  
Coba continue 2  
Coba continue 3  
Coba continue 5
```

### C. Struktur Pengulangan Bersarang

Di dalam sebuah struktur pengulangan diperbolehkan untuk membuat pengulangan lainnya. Dengan kata lain, pengulangan di dalam pengulangan. Pada operasi FOR dalam FOR, FOR yang paling dalam akan diselesaikan lebih dahulu, baru dieksekusi FOR yang di luar. Sehingga instruksi-instruksi yang didapat pada FOR yang paling dalam akan paling banyak dieksekusi.

#### **Algoritma** Persegi\_Panjang

Deskripsi

`i,j: integer`

Deklarasi

`for i ← 1 to 5 do`

`for j ← 1 to 9 do`

`Write('#')`

`endfor`

`writeln`

`endfor`

**Program Persegi Panjang**

```
{Program Cetak Persegi Panjang '#' Full
{ 1}Program Contoh73;
{ 2}var
{ 3}  i,j:integer;
{ 4}begin
{ 5}  for i:=1 to 5 do
{ 6}  begin
{ 7}      for j:=1 to 9 do
{ 8}          write('#':2);
{ 9}      writeln;
{10}  end;
{11}end.
```

**Catatan:**

1. Bagaimana output program Contoh73 di atas? ya, outputnya adalah persegi panjang '#' dengan ukuran 5×9.
2. Bagaimana outputnya kalau writeln pada baris 9 dihapus?
3. Coba pikirkan untuk memodifikasi program Contoh73 sedemikian sehingga karakter yang digunakan dan ukuran persegi panjangnya terserah pengguna? ya...tentunya kita harus menambahkan 3 variabel dan juga statemen input untuk karakter, nilai maksimal untuk i dan j sehingga programnya menjadi Contoh74 di bawah.

```
{Program Cetak Persegi Panjang Full}
{ 1}Program Contoh74;
{ 2}var
{ 3}   i,j,m,n:integer;
{ 4}   kar:char;
{ 5}begin
{ 6}   Write('Karakter yang digunakan: ');
{ 7}   readln(kar);
{ 8}   writeln('Ukuran persegi panjang yang diinginkan');
{ 9}   write('Panjang= ');readln(m);
{10}   write('Lebar=   ');readln(n);
{11}   for i:=1 to m do
{12}   begin
{13}       for j:=1 to n do
{14}           write(kar:2);
{15}       writeln;
{16}   end;
{17}end.
```

Kemudian bagaimana caranya untuk memodifikasi program Contoh74 di atas agar outputnya menjadi persegi panjang yang tengahnya kosong?

Contoh output bagian utamanya (4×7):

```
# # # # # # #
#           #
#           #
# # # # # # #
```

Programnya menjadi:

```
{Program Cetak Persegi Panjang Kosong}
{ 1}Program Contoh75;
{ 2}var
{ 3}  i,j,m,n:integer;
{ 4}  kar:char;
{ 5}begin
{ 6}  Write('Karakter yang digunakan: ');
{ 7}  readln(kar);
{ 8}  writeln('Ukuran persegi panjang yang diinginkan');
{ 9}  write('Panjang= ');readln(m);
{10}  write('Lebar=   ');readln(n);
{11}  for i:=1 to m do
{12}  begin
{13}      for j:=1 to n do
{14}          if (i=1)or(i=m)or(j=1)or(j=n) then
{15}              write(kar:2)
{16}          else write(' ':2);
{17}      writeln;
{18}  end;
{19}end.
```

#### D. Penugasan

1. Buatlah program untuk mencetak output seperti di bawah ini dimana yang diinput hanya jumlah barisnya saja.

0 1 2 3 4 5 6 7 8 9

9 8 7 6 5 4 3 2 1 0

0 1 2 3 4 5 6 7 8 9

9 8 7 6 5 4 3 2 1 0

0 1 2 3 4 5 6 7 8 9

2. Modifikasi Contoh74 sedemikian sehingga outputnya menjadi:

- a. segitiga rata kiri
- b. segitiga rata kanan
- c. segitiga rata tengah

3. Modifikasi Contoh75 sedemikian sehingga outputnya menjadi:

- d. segitiga rata kiri
- e. segitiga rata kanan
- f. segitiga rata tengah

4. Buatlah program yang outputnya segitiga pascal.

## MODUL 8 TIPE DATA TERBILANG DAN SUB RANGE SERTA SUB PROGRAM 1

---

### 8.1 Deskripsi Singkat

Pada modul 8 ini akan dipelajari materi tentang tipe data enumerated(terbilang) dan sub range serta sub program di Pascal yaitu fungsi dan prosedur. Modul 8 ini merupakan materi yang penting jika program yang dibuat sudah menjadi kompleks.

### Tujuan Praktikum

Setelah praktikum pada modul 8 ini mahasiswa diharapkan memahami dengan baik tipe data terbilang dan sub range serta fungsi dan prosedur, dapat membedakan antara fungsi dan prosedur dan bisa mengimplementasikannya dalam program Pascal.

### Material Praktikum

Kegiatan pada modul 8 ini memerlukan material berupa materi singkat serta program Pascal

### Kegiatan Praktikum

#### A. Konsep dan Bentuk Umum Tipe Data Terbilang dan Sub Range

Tipe data terbilang dan sub range adalah tipe data ordinal yang dapat didefinisikan oleh pemrogram sesuai dengan keperluannya. Dengan menggunakan tipe terbilang, pemrogram dapat mendefinisikan suatu tipe data yang nilainya belum ada sebelumnya. sedangkan tipe sub range, dapat digunakan untuk mengambil sebagian nilai dari tipe data yang sudah didefinisikan sebelumnya.

##### 1. Terbilang

Deklarasi:

- a. tipe  
`type Namatype=(nilai_1,nilai_2,...,nilai_n);`
- b. variabel  
`var Namavar: (nilai_1,nilai_2,...,nilai_n);`

Contoh:

```
type hari=( ahad, senin, selasa, rabu, kamis, jumat, sabtu);  
var kelas: ( _1ks1, _1ks2, _1ks3, _1ks4, _1ks5);
```

##### 2. Sub Range

Deklarasi:

- a. Tipe  
`type namatype = batas_bawah .. batas_atas;`
- b. var

var namavar : batas\_bawah.. batas\_atas;

Contoh:

type nilai\_ujian= 0..100;

var grade : 'A' .. 'E';

Contoh Program:

```
{ 1}Program Contoh81;
{ 2}uses crt;
{
{ 3}type
Harilpekan=(Ahad,Senin,Selasa,Rabu,Kamis,Jumat,Sabtu);
{ 4}    haribulan=1..31;
{ 5}
{ 6}var i,awalbulan:harilpekan;
{ 7}    j,tglmaks:haribulan;
{ 8}    x,y,k,lebar:integer;
{ 9}
{10}begin
{11}    clrscr;
{12}    write('Hari pertama awal bulan: ');
{13}    readln(awalbulan);
{14}    write('Jumlah hari bulan ini: ');
{15}    readln(tglmaks);writeln;
{16}
{17}    lebar:=7;
{18}    for i:=Ahad to sabtu do
{19}        write(i:lebar);
{20}
{21}    y:=5;
{22}    x:=ord(awalbulan);
{23}    for j:=1 to tglmaks do
{24}    begin
{25}        if x=7 then
{26}        begin
{27}            x:=0;
{28}            y:=y+1;
{29}        end;
{30}        gotoxy(lebar*x+1,y);write(j:3);
{31}        x:=x+1;
{32}    end;
{33}    readln
{34}end.
```

keterangan:

- 1) Pada baris ke-2 dideklarasikan unit CRT, dimana di dalam unit ini prosedur clrscr dan gotoxy yang digunakan pada program didefinisikan. apa yang terjadi jika baris ke-2 tersebut dijadikan sebagai komentar?
- 2) Baris ke-17 dan 19 adalah untuk menampilkan judul kolom.
- 3) output pada baris2 selanjutnya di atur dengan menggunakan procedure gotoxy sehingga tidak perlu menggunakan writeln.
- 4) Baris ke-21 dan 22 adalah untuk mempersiapkan posisi penulisan tanggal.
- 5) Di baris ke-30 pada argumen pertama dari prosedur gotoxy, tertulis lebar\*x+1. Apa tujuan dari +1 tersebut? apa yang terjadi jika +1 dihapus?Kenapa outputnya jadi aneh? (petunjuk: nilai x minimal adalah 1)

## B. Konsep dan Bentuk Umum Fungsi dan Prosedur

Sub program atau sering disebut sebagai program kecil merupakan bagian penting dari sebuah program yang berfungsi untuk membantu programmer dalam melakukan efisiensi dengan cara menghindari penulisan kode yang berulang-ulang.

Sub program pada Pascal memiliki begin dan end nya sendiri dan dapat pula memiliki variabelnya sendiri (variabel lokal). Variabel yang dideklarasikan pada sub program tidak dapat digunakan pada program utama. Variabel yang dideklarasikan pada program utama disebut dengan variabel global.

Sub Program pada Pascal dibagi menjadi 2 macam yaitu procedure (prosedur) dan function (fungsi). Sub program dideklarasikan pada bagian kamus program.

### 1. Fungsi

```
Function NamaFungsi[(parameter)]: tipe_fungsi;  
[deklarasi variabel];  
begin  
    statemen-statemen;  
    [NamaFungsi:=nilai_fungsi]  
end;
```

Contoh tanpa parameter:

```
Program Belajar_fungsi;  
  
var n,hasil:integer;
```



```

Function Pangkat2:integer;
begin
    Pangkat2:=n*n;
end;
begin
    write(' N= ');readln(n);
    hasil:=pangkat2;
    writeln(' Pangkat 2 dari ',n,' = ',hasil);
    readln
end.

```

Contoh dengan parameter:

```

Program Belajar_fungsi;

var n,hasil:integer;

Function Pangkat2(a:integer):integer;
begin
    Pangkat2:=a*a;
end;
begin
    write(' N= ');readln(n);
    hasil:=pangkat2(n);
    writeln(' Pangkat 2 dari ',n,' = ',hasil);
    readln
end.

```

## 2. Prosedur

```

Procedure NamaProsedure[(parameter)];
[deklarasi variabel];

```

```
begin
    statemen-statemen;
end;
```

**Contoh tanpa parameter:**

```
Program Belajar_Prosedur;

var n,hasil:integer;

    Procedure Pangkat2;
    begin
        hasil:=n*n;
    end;
begin
    write(' N= ');readln(n);
    pangkat2;
    writeln(' Pangkat 2 dari ',n,' = ',hasil);
    readln
end.
```

**Contoh dengan parameter:**

```
Program Belajar_Prosedur;

var n,hasil:integer;

    Procedure Pangkat2(a:integer);
    begin
        hasil:=a*a;
    end;
begin
    write(' N= ');readln(n);
    pangkat2(n);
```

```
writeln(' Pangkat 2 dari ',n,' = ',hasil);  
readln  
end.
```

Hasil dari keempat program di atas persis sama yaitu: (misal kita inputkan nilai n sama dengan 3)

```
N= 3  
Pangkat 2 dari 3= 9
```

#### Catatan:

- 1) Fungsi dan prosedur baik dengan parameter atau tidak dapat menyelesaikan persoalan yang sama.
- 2) Tentu ada perbedaan antara fungsi dan prosedur. Fungsi mengembalikan sebuah nilai karenanya dalam pendeklarasiannya fungsi harus memiliki tipe dan dibagian akhir ada statemen yang memberikan nilai pada fungsi tersebut. sedangkan prosedur tidak mengembalikan nilai.
- 3) penggunaan parameter dapat menambah fleksibilitas dari fungsi atau prosedur yang dibuat.

#### Penugasan

- 1) Modifikasi program Contoh81 agar bisa menampilkan kalender 1 tahun.
- 2) Modifikasi program yang ada di penugasan pada modul 7 dengan menggunakan sub program dengan fungsi dan prosedur serta dengan dan tanpa parameter.
- 3) Buat program dengan kalkulator dengan fungsi KABATAKU(kali, bagi, tambah, kurang) dimana saat program dijalankan, yang pertama muncul adalah menu seperti di bawah ini:

```
Selamat datang di Kalkulator Sederhana  
Silahkan pilih menu berikut:  
1. Penjumlahan  
2. Pengurangan  
3. Perkalian  
4. Pembagian  
5. Keluar  
Pilihan Anda:
```

untuk menu 1-4,buat dengan menggunakan sub program.

## MODUL 9 SUB PROGRAM 2

---

### 9.1 Deskripsi Singkat

Pada modul 9 ini akan dipelajari tentang jangkauan pengenalan/identifikasi dan transfer parameter. pemahaman terhadap kedua hal ini menjadi sangat penting dalam pembuatan program tingkat lanjut.

### Tujuan Praktikum

Setelah praktikum pada modul 9 ini diharapkan:

- 1) mahasiswa memahami jangkauan identifikasi dan dapat menggunakannya dalam program
- 2) mahasiswa memahami transfer parameter dan dapat menggunakannya dalam program

### Material Praktikum

Kegiatan pada modul 9 ini memerlukan material berupa program Pascal

### Kegiatan Praktikum

#### A. Identifier

**Identifier lokal** adalah identifier yang dideklarasikan di dalam sub program, selain itu adalah **identifier global**. Identifier lokal hanya dapat digunakan lokal di dalam sub program dimana identifier tersebut dideklarasikan, sedangkan identifier global dapat digunakan dimana saja sepanjang sudah dideklarasikan, termasuk di dalam suatu sub program sepanjang tidak ada identifier dengan nama yang sama.

Contoh Program:

```
{Contoh program yang menunjukkan identifier lokal yang berlaku di dalam sub program tersebut saja}
```

```
Program Contoh91;
```

```
var x,y:integer;
```

```
    procedure subpro(a:integer);
```

```
    var b:integer;
```

```
    begin
```

```

        b:=a;
        writeln(b);
    end;

begin
    write(' Input nilai a= ');
    readln(a);
    subpro(a);
end.

```

saat program di compile, pesan kesalahan yang muncul:

```

1 { Contoh program yang menunjukkan identifier lokal yang berlaku di dalam
2 sub program tersebut saja}
3
4 Program Contoh92;
5 var x,y:integer;
6
7 procedure subpro(a:integer);
8 var b:integer;
9 begin
10     b:=a;
11     writeln(b);
12 end;
13
14 begin
15     write(' Input nilai a= ');
16     readln(a);
17     subpro(a);
18 end.

```

Compilation failed due to following error(s).

```

Copyright (c) 1993-2012 by Florian Klaempfl and others
Target OS: Linux for x86-64
Compiling main.pas
main.pas(16,13) Error: Identifier not found "a"
main.pas(17,13) Error: Identifier not found "a"
main.pas(18,4) Fatal: There were 2 errors compiling module, stopping
Fatal: Compilation aborted

```

terlihat error muncul pada baris ke 16 dan 17, dimana variabel a tidak dikenali di program utama walau sudah dideklarasikan di prosedur subpro.

```

{Contoh program yang menunjukkan identifier lokal yang
berlaku di dalam sub program tersebut saja}

```

```

Program Contoh92;
var x,y:integer;

    procedure subpro1(a:integer);
    var b:integer;

```

```

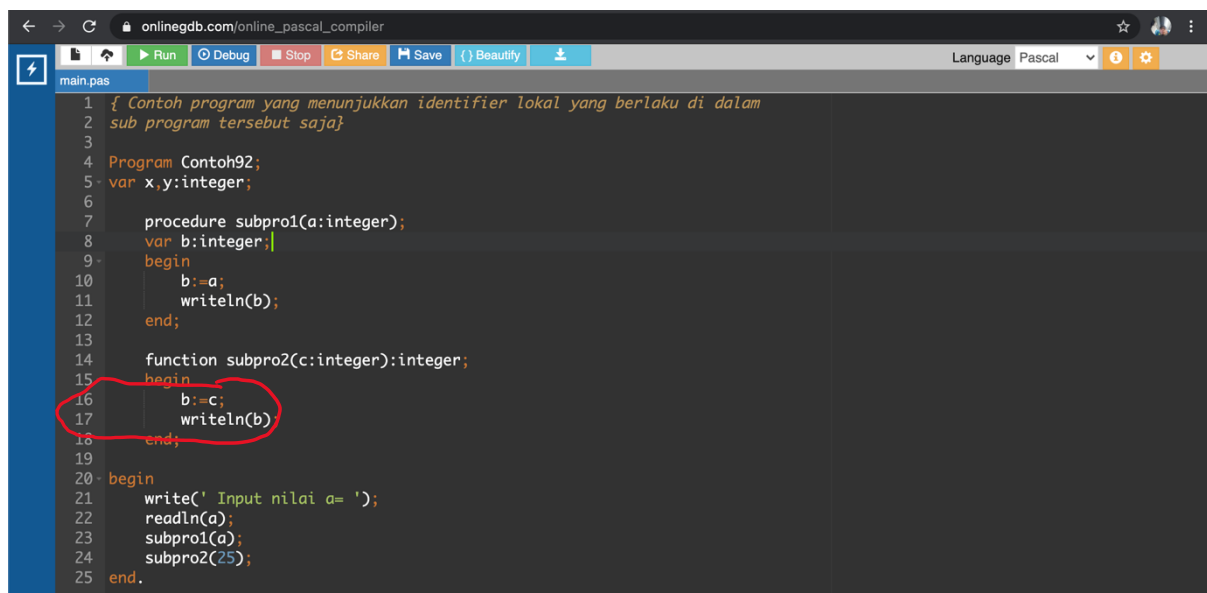
begin
    b:=a;
    writeln(b);
end;

function subpro2(c:integer):integer;
begin
    b:=c;
    writeln(b);
end;

begin
    write(' Input nilai a= ');
    readln(a);
    subpro1(a);
    subpro2(25);
end.

```

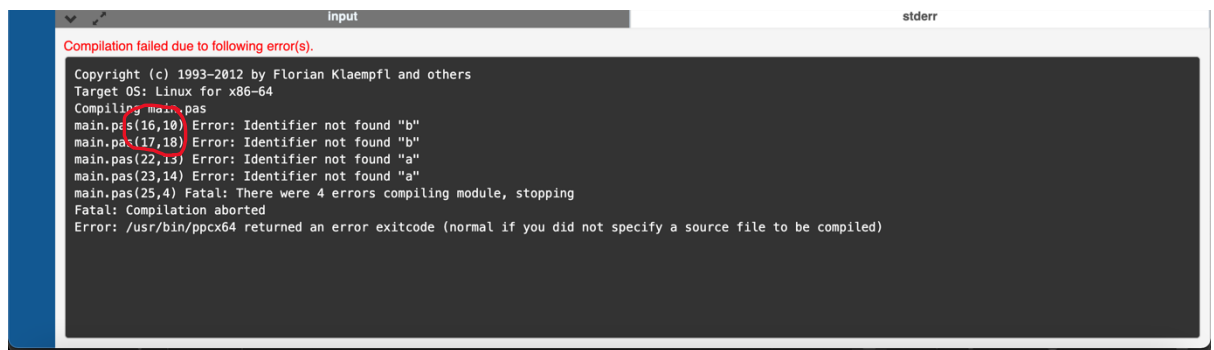
saat program dicompile, hasilnya seperti di bawah ini:



```

1 { Contoh program yang menunjukkan identifer lokal yang berlaku di dalam
2 sub program tersebut saja}
3
4 Program Contoh92;
5 var x,y:integer;
6
7 procedure subpro1(a:integer);
8 var b:integer;
9 begin
10     b:=a;
11     writeln(b);
12 end;
13
14 function subpro2(c:integer):integer;
15 begin
16     b:=c;
17     writeln(b);
18 end;
19
20 begin
21     write(' Input nilai a= ');
22     readln(a);
23     subpro1(a);
24     subpro2(25);
25 end.

```



```
Input stderr
Compilation failed due to following error(s).
Copyright (c) 1993-2012 by Florian Klaempfl and others
Target OS: Linux for x86-64
Compiling 'main.pas'
main.pas(16,10) Error: Identifier not found "b"
main.pas(17,18) Error: Identifier not found "b"
main.pas(22,13) Error: Identifier not found "a"
main.pas(23,14) Error: Identifier not found "a"
main.pas(25,4) Fatal: There were 4 errors compiling module, stopping
Fatal: Compilation aborted
Error: /usr/bin/ppcx64 returned an error exitcode (normal if you did not specify a source file to be compiled)
```

Terlihat pada baris 16 dan 17 muncul error yg disebabkan variabel b tidak dikenali di fungsi subpro2 walau sudah dideklarasikan pada prosedur subpro1.

{ Contoh program yang menunjukkan identifier global dapat digunakan baik di dalam sub program atau di program utama setelah dideklarasikan}

Program Contoh93;

var x,y:integer;

procedure subpro(a:integer);

begin

x:=a+4;

writeln(' x= ',x);

end;

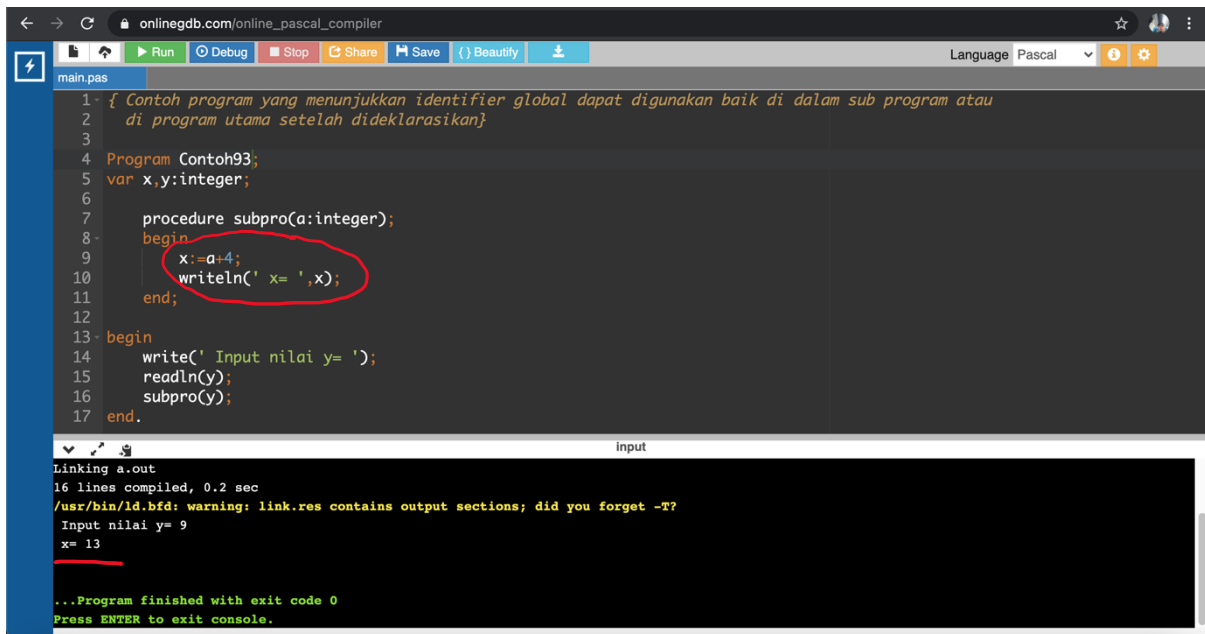
begin

write(' Input nilai y= ');

readln(y);

subpro(y);

end.



```
1 { Contoh program yang menunjukkan identifier global dapat digunakan baik di dalam sub program atau
2 di program utama setelah dideklarasikan}
3
4 Program Contoh93;
5 var x,y:integer;
6
7 procedure subpro(a:integer);
8 begin
9     x:=a+4;
10    writeln(' x= ',x);
11 end;
12
13 begin
14     write(' Input nilai y= ');
15     readln(y);
16     subpro(y);
17 end.
```

Input

```
Linking a.out
16 lines compiled, 0.2 sec
/usr/bin/ld.bfd: warning: link.res contains output sections; did you forget -T?
Input nilai y= 9
x= 13

...Program finished with exit code 0
Press ENTER to exit console.
```

di dalam prosedur subpro pada baris 9 dan 10 dapat digunakan variabel x yang sudah dideklarasikan di luar prosedur tersebut dan program tidak error saat di running sehingga terlihat outputnya.

```
{ Contoh program yang menunjukkan identifier global dapat
digunakan baik di dalam sub program atau di program utama
setelah dideklarasikan}
```

```
Program Contoh94;
```

```
    procedure subpro(a:integer);
    begin
        x:=a+4;
        writeln(' x= ',x);
    end;
```

```
var x,y:integer;
```

```
begin
```

```
    write(' Input nilai y= ');
    readln(y);
    subpro(y);
```

```
end.
```



```
1: { Contoh program yang menunjukkan identifier global dapat digunakan baik di dalam sub program atau
2:   di program utama setelah dideklarasikan}
3:
4: Program Contoh94;
5:
6:   procedure subpro(a:integer);
7:   begin
8:     x:=a+4;
9:     writeln(' x= ',x);
10:  end;
11:
12: var x,y:integer;
13: begin
14:   write(' Input nilai y= ');
15:   readln(y);
16:   subpro(y);
17: end.
```

Compilation failed due to following error(s).

```
Free Pascal Compiler version 2.6.2-8 [2014/01/22] for x86_64
Copyright (c) 1993-2012 by Florian Klaempfl and others
Target OS: Linux for x86_64
Compiling main.pas
main.pas(8,10) Error: Identifier not found "x"
main.pas(9,25) Error: Identifier not found "x"
main.pas(17,4) Fatal: There were 2 errors compiling module, stopping
Fatal: Compilation aborted
```

saat variabel x dideklarasikan di bawah prosedur maka muncul error yg menyatakan variabel x (baris 8 dan 9) tidak dikenali di dalam prosedur.

{ Contoh program yang menunjukkan identifier global dapat digunakan di dalam sub program selama tidak ada nama yang sama}

Program Contoh95;

var x,y:integer;

procedure subpro(a:integer);

var x:char;

begin

x:='C';

y:=a+4;

writeln('Nilai di dalam subpro: x= ',x,' y= ',y);

end;

begin

x:=6;

y:=10;

subpro(y);

writeln('Nilai di program utama: x= ',x,' y= ',y);

end.

```
1 { Contoh program yang menunjukkan identifier global dapat digunakan di dalam sub program selama tidak ada nama yang sama}
2 Program Contoh95;
3 var x,y:integer;
4
5 procedure subpro(a:integer);
6   var x:char;
7   begin
8     x:='C';
9     y:=a+4;
10    writeln('Nilai di dalam subpro: x= ',x,' y= ',y);
11  end;
12
13 begin
14   x:=6;
15   y:=10;
16   subpro(y);
17   writeln('Nilai di program utama: x= ',x,' y= ',y);
18 end.
```

input

```
/usr/bin/ld.bfd: warning: link.res contains output sections; did you forget -T?
17 lines compiled 0.1 sec
Nilai di dalam subpro: x= C y= 14
Nilai di program utama: x= 6 y= 14
...Program finished with exit code 0
Press ENTER to exit console.
```

terlihat variabel x dideklarasikan sebagai global (baris 3) dan local (baris 6) identifier sedangkan y sebagai global identifier saja. kemudian saat di running, didapat hasil nilai x yg ditampilkan di subpro adalah C, sedangkan saat di program utama nilainya 6 sedangkan y nilainya sama2 14 (lingkaran biru).

## B. Transfer Parameter

Transfer parameter terbagi 2, yaitu tranfer by value ketika parameternya saat dideklarasikan tanpa kata kunci var dan transfer by reference/location jika dideklarasikan dengan var.

Contoh Program:

```
{Contoh program yang menunjukkan perbedaan transfer
parameter By Value dan By Location}
Program Contoh96;
var x,y:integer;

procedure subpro(a:integer;var b:integer);
var c:integer;
begin
  c:=a; a:=b; b:=c;
end;
```

```

begin
    x:=5; y:=23;
    subpro(x,y);
    writeln('transfer by value:    x= ',x);
    writeln('transfer by location: y= ',y);
end.

```

```

1 { Contoh program yang menunjukkan perbedaan transfer parameter By Value dan By Location}
2 Program Contoh96;
3 var x,y:integer;
4
5 procedure subpro(a:integer;var b:integer);
6 var c:integer;
7 begin
8   c:=a; a:=b; b:=c;
9 end;
10
11 begin
12   x:=5; y:=23;
13   subpro(x,y);
14   writeln('transfer by value:    x= ',x);
15   writeln('transfer by location: y= ',y);
16 end.

```

input

```

/usr/bin/ld.bfd: warning: link.res contains output sections; did you forget -T?
16 lines compiled, 0.0 sec
transfer by value:    x= 5
transfer by location: y= 5
...Program finished with exit code 0
Press ENTER to exit console.

```

Terlihat pada program di atas bahwa parameter aktual x ditransfer by value ke parameter formal a sedangkan parameter aktual y ditransfer by location ke parameter formal b sehingga hasilnya, nilai x tetap tidak berubah sedangkan y nilainya berubah menjadi nilai x sebagaimana dilakukan di dalam prosedur subpro.

## Penugasan

- 1) Buatlah program yang menampilkan hasil dari penjumlahan 2 buah bilangan dengan menggunakan prosedur dengan 2 parameter yang merupakan bilangan yang akan dijumlahkan dan tampilkan hasil penjumlahannya di dalam program utama.
- 2) Buatlah program seperti soal no. 1 tetapi dengan tambahan 1 parameter yang menyimpan nilai dari hasil penjumlahan.

**catatan:** soal nomor 1 dan 2 yang lebih pas jika dibuat dengan menggunakan fungsi dengan 2 parameter yg merupakan bilangan yang akan dijumlahkan dan fungsinya menyimpan hasil penjumlahan kedua bilangan tersebut.

- 3) Buatlah program kalkulator kabataku yang saat program dijalankan, yang muncul pertama kali adalah menu kabataku dan yg ke-5 adalah keluar

[program], dimana masing masing dari kabataku dibuat dalam sebuah sub program sendiri.

- 4) Buatlah program yang menghitung Faktorial, Permutasi dan Kombinasi dimana pada subprogram permutasi dalam proses penghitungannya menggunakan sub program faktorial dan sub program kombinasi menggunakan permutasi.

## MODUL 10 ARRAY

---

### 10.1 Deskripsi Singkat

Pada modul 10 ini akan dibahas penggunaan array. Penggunaan array salah satunya bertujuan untuk efisiensi penyimpanan dan pengambilan data.

### 10.2 Tujuan Praktikum

Setelah praktikum pada modul 10 ini diharapkan:

1. Mahasiswa memahami dan menguasai struktur data array
2. Mahasiswa dapat menggunakan struktur data array dengan terampil pada program.

### 10.3 Material Praktikum

Kegiatan pada modul 10 ini memerlukan material berupa program Pascal

### 10.4 Kegiatan Praktikum

**Array** atau **larik** adalah suatu struktur data berindeks yang dapat digunakan untuk menyimpan sejumlah variabel yang memiliki tipe data yang sama.

Misalkan program kita akan menggunakan 10 buah data integer. Jika kita tidak menggunakan array, maka kita perlu mendeklarasikan 10 buah variabel dengan tipe integer seperti contoh berikut:

```
Var  
  x1, x2, x3, x4, x5, x6, x7, x8, x9, x10 : integer;
```

Namun jika kita menggunakan array, maka kita cukup mendeklarasikan satu array yang berisi 10 data integer seperti contoh berikut:

```
Var  
  X : array[1..10] of integer;
```

Dengan kata lain, untuk menyimpan 10 elemen variabel bertipe integer, kita cukup menggunakan satu identifier X.

Array merupakan sebuah struktur data yang statis. Statis artinya ketika *runtime* ukurannya tidak dapat berubah.

Deklarasi umum dari array adalah:

```
NamaArray : array[IndeksAwal..IndeksAkhir] of tipe_data;
```

Nilai IndeksAwal harus lebih kecil atau sama dengan nilai IndeksAkhir. Kita bisa menggunakan tipe data yang memiliki nilai ordinal sebagai IndeksAwal dan IndeksAkhir. Yang banyak digunakan sebagai indeks array adalah bilangan bulat.

Contoh deklarasi array:

```
Var
A: array[0..9] of Integer;
B: array[10..20] of String;
C: array['a'..'j'] of Boolean;
X : array[1..100] of Integer;
```

Untuk mengakses setiap elemen array, kita menggunakan indeksny. Misal pada deklarasi variabel X di atas, indeks array dimulai dari 1 sampai 100 yang nantinya dituliskan pada kurung siku [ ]. Misalnya:

```
...
X[1] := 10; {array X indeks pertama kita isi nilai 10}
X[2] := X[1] - 5; {array X indeks kedua kita isi nilai
array X indeks pertama dikurangi 5 yang mana hasilnya
adalah 5}
X[3] := X[2] + X[1];
Writeln(X[3]); {output = 15}
...
```

Kita tidak bisa mengakses indeks array di luar range yang telah dideklarasikan karena akan menyebabkan error *out of range*. Misalnya:

```
Var
  X: array[1..5] of integer;
Begin
  X[6] := 15 {error, indeks yang akan diakses di luar
range}
```

Penggunaan array bertujuan untuk efisiensi manajemen data dengan memanfaatkan indeks array. Berikut adalah contoh pencarian nilai terbesar dari 3 bilangan tanpa menggunakan array:

```
if x > y then
  if x > z then

    terbesar := x
```

```

    else

        terbesar := z
else
    if y > z then

        terbesar := y
else
    terbesar := z;

```

Bagaimana jika bilangannya ada 100? Atau ada 1000? Jika kita menggunakan array, maka berapapun jumlah datanya tidak menjadi masalah. Perhatikan program berikut:

```

Var
    X : array[1..10] of Integer;
    Terbesar: Integer;
    i : Integer;
Begin
    For i:=1 to 10 do
        Begin
            Write('Input data ke-', i, '= ');
            Readln(X[i]);
        End;
    {di awal dimisalkan bahwa data terbesar adalah X[1]}
    Terbesar := X[1];
    For i:=2 to 10 do {perulangan bergerak dari indeks ke 2
                      sampai 10}
        Begin
            If X[i] > Terbesar then Terbesar := X[i];
        End;
    Writeln(Terbesar); {menampilkan data terbesar yang
                        diinput user}
    Readln;
End.

```

18:59 ...139KB/d

```
0 program simple_array;
1 uses crt;
2 var
3   x : array[1..10] of integer;
4 begin
5   x[1]:=10;
6   x[2]:=22;
7   x[3]:=45;
8
9   writeln('nilai x1=', x[1]);
10  writeln('nilai x2=', x[2]);
11  writeln('nilai x3=', x[3]);
12  readln;
13 end.
14
```

→| := ; . [ ] : ' ( )

< GIF ⚙️ 🔍 ...

1 2 3 4 5 6 7 8 9 0

@ # \$ % & \* + ( ) /

=\< \* " ' : ; ! ? <X>

ABC , 12 34 Google . ←

19:00 ...183KB/d

← simple\_array.pas

```
Initialize the console screen...
Size: 45x49
-----
execute file: /storage/emulated/0/PascalCompiler/
simple_array.pas
-----

nilai x1=10
nilai x2=22
nilai x3=45
```

1 2 3 4 5 6 7 8 9 0

q w e r t y u i o p

a s d f g h j k l

↑ z x c v b n m <X>

?123 , Google . ←

19:06 ...72,1KB/d

```
0 program data_terbesar;
1 uses crt;
2 var
3   x:array[1..4] of integer = (1, 4, 16, 12);
4   i terbesar: integer;
5 begin
6   terbesar:=x[1];
7   for i:=2 to 4 do
8   begin
9     if x[i] > terbesar then terbesar:=x[i];
10    end;
11    writeln('terbesar = ', terbesar);
12    readln;
13 end.
14
```

→| := ; . [ ] : ' ( )

< GIF ⚙️ 🔍 ...

1 2 3 4 5 6 7 8 9 0

@ # \$ % & \* + ( ) /

=\< \* " ' : ; ! ? <X>

ABC , 12 34 Google . ←

19:06 ...185KB/d

← data\_terbesar.pas

```
Initialize the console screen...
Size: 45x49
-----
execute file: /storage/emulated/0/PascalCompiler/
data_terbesar.pas
-----

terbesar = 16
```

1 2 3 4 5 6 7 8 9 0

q w e r t y u i o p

a s d f g h j k l

↑ z x c v b n m <X>

?123 , Google . ←



## A. Array 2 Dimensi

Pembahasan sebelumnya adalah untuk array satu dimensi. Array yang memiliki dimensi lebih dari 1 disebut array multidimensi. Array yang memiliki 2 dimensi disebut dengan array 2 dimensi.

Setiap elemen array dapat diakses menggunakan **indeks[baris, kolom]**

Misal kita mendeklarasikan array 2 dimensi sebagai berikut:

```
Var
  X : array[1..3, 1..4] of integer;
```

Maka jumlah data yang disimpan pada variabel X adalah sebanyak baris kali kolom atau  $3 \times 4 = 12$ . Kira-kira memori akan menyimpan variabel X seperti berikut:

<b>X</b>				
Baris/Kolom	1	2	3	4
1	[1,1]	[1,2]	[1,3]	[1,4]
2	[2,1]	[2,2]	[2,3]	[2,4]
3	[3,1]	[3,2]	[3,3]	[3,4]

Setiap elemen array dapat diakses menggunakan **indeks[baris, kolom]**. Misalnya:

```
Var
  X : array[1..3, 1..4] of integer;
Begin
  X[1,3] := 10;
  Writeln(X[1,3]); {output 10}
End.
```

Array 2 dimensi banyak digunakan untuk operasi matriks. Perhatikan listing program untuk penjumlahan matriks berikut ini:

```
Program penjum_matriks;
var
  a,b,c: array[1..3, 1..3] of integer;
  i,j: integer;
begin
  writeln('buat matriks A');
  for i:=1 to 3 do
    for j:=1 to 3 do
```

```

        begin
            write([' ', i, ' ', ' ', j, ' '=]);
            readln(a[i, j]);
        end;
writeln('buat matriks B');
for i:=1 to 3 do
    for j:=1 to 3 do
        begin
            write([' ', i, ' ', ' ', j, ' '=]);
            readln(b[i, j]);
        end;

writeln('Matriks A');
for i:=1 to 3 do
begin
    for j:=1 to 3 do
        write(a[i, j], ' ');
    writeln;
end;
writeln('Matriks B');
    for i:=1 to 3 do
begin
    for j:=1 to 3 do
        write(b[i, j], ' ');
    writeln;
end;
writeln('matriks C');
for i:=1 to 3 do
begin
    for j:=1 to 3 do
        begin
            c[i, j]:=a[i, j]+b[i, j];
            write(c[i, j], ' ');
        end;
    writeln;
end;

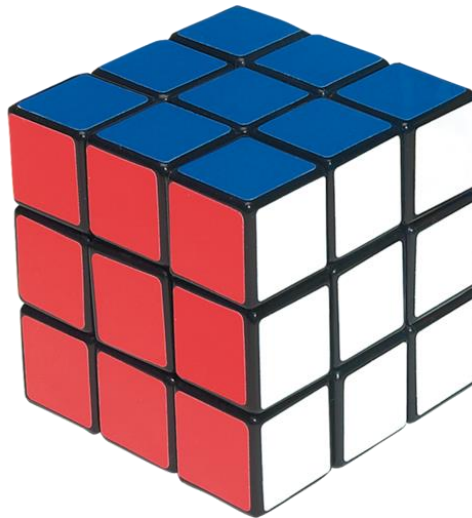
```

```
end;  
readln;  
end.
```

Program di atas menginput matriks A, menginput matriks B, kemudian menampilkan matriks A dan matriks B kemudian menampilkan matriks C sebagai hasil penjumlahan dari matriks A+B.

### C. Array Multi Dimensi

Array dapat kita deklarasikan dalam sejumlah dimensi tergantung kebutuhan kita. Misal sebuah array yang memiliki 3 dimensi dengan masing-masing dimensinya adalah 3, dapat digambarkan seperti sebuah rubric cube.



Rubric Cube adalah array 3x3x3

Contoh deklarasinya adalah sebagai berikut:

```
Var  
X : array[1..3, 1..3, 1..3] of integer;
```

Maka variabel X memiliki elemen sebanyak  $3 \times 3 \times 3 = 27$  buah. Untuk mengakses tiap elemen, indeksinya sebanyak 3 seperti contoh berikut ini.

```
Var  
X : array[1..3, 1..3, 1..3] of integer;  
Begin  
  X[1,1,1] := 10;  
  Writeln(X[1,1,1]); {output 10}
```

End.

Terdapat 27 elemen pada X yaitu:

```
X[1,1,1] , X[1,2,1] , X[1,3,1], X[1,1,2] , X[1,2,2] , X[1,3,2],  
X[1,1,3] , X[1,2,3] , X[1,3,3], X[2,1,1] , X[2,2,1] , X[2,3,1],  
X[2,1,2] , X[2,2,2] , X[2,3,2], X[2,1,3] , X[2,2,3] , X[2,3,3],  
X[3,1,1] , X[3,2,1] , X[3,3,1], X[3,1,2] , X[3,2,2] , X[3,3,2],  
X[3,1,3] , X[3,2,3] , X[3,3,3]
```

Dapatkan anda bayangkan array 4 dimensi, 5 dimensi, dan seterusnya?

Array dapat juga kita nyatakan pada bagian `type`, untuk selanjutnya kita gunakan pada bagian `var`. Perhatikan contoh berikut:

Type

```
Matrix = array[1..3, 1..3] of real;
```

Var

```
A,B,C : Matrix;
```

Untuk array multidimensi terdapat cara lain untuk mengakses indeks elemen array selain menggunakan koma yaitu menggunakan kurung siku terpisah. Misalnya:

Var

```
X : array[1..3, 1..4] of integer;
```

Begin

```
X[1][3] := 10;
```

```
Writeln(X[1][3]); {output 10}
```

End.

## 10.5 Penugasan

1. Buatlah program statistik sederhana yang memiliki fungsi input data, menampilkan semua data, menampilkan bilangan terkecil, bilangan terbesar, dan rata-rata. Fungsi-fungsi tersebut dibuat dengan menggunakan sub program sebagaimana yang telah dipelajari sebelumnya.

Selamat datang di aplikasi statistik sederhana

Silahkan pilih menu berikut:

1. Input data
2. Tampilkan semua data
2. Tampilkan data terkecil
3. Tampilkan data terbesar
4. Tampilkan rata-rata
5. Keluar

2. Buatlah program untuk:

- Melakukan perkalian matriks
- Menghitung determinan matriks
- Melakukan transpose matriks
- Melakukan invers matriks

## MODUL 11 RECORD

---

### 11.1 Deskripsi Singkat

Pada modul 11 ini akan dibahas penggunaan record.

### 11.2 Tujuan Praktikum

Setelah praktikum pada modul 11 ini diharapkan:

1. Mahasiswa memahami dan menguasai struktur data record
2. Mahasiswa dapat menggunakan struktur data record dengan terampil pada program.

### 11.3 Material Praktikum

Kegiatan pada modul 11 ini memerlukan material berupa program Pascal

### 11.4 Kegiatan Praktikum

Berbeda dengan array yang berfungsi untuk menyimpan sejumlah data dengan tipe yang sama, record merupakan sebuah variabel untuk menyimpan data dengan tipe yang bisa berbeda.

Record dapat juga dideklarasikan pada bagian type untuk selanjutnya kita gunakan pada bagian var.

```
Type
Identifier = Record
    Namafield-1 : Type
    Namafield-2 : Type
    .....
    Namafield-N : Type
End
Var
    Identifier : Record
    Namafield-1 : Type
    Namafield-2 : Type
    .....
    Namafield-N : Type
End
```

Contoh:

```

type
    data_mahasiswa = record
        nim : string[10];
        nama : string[25];
        alamat : string[20];
        nilai : longint;
    end;
var
    mahasiswa : data_mahasiswa;

```

atau bisa juga langsung kita deklarasikan pada bagian var

```

var
    mahasiswa : record
        nim : string[10];
        nama : string[25];
        alamat : string[20];
        nilai : longint;
    end;

```

Untuk mengakses elemen record kita menggunakan nama identifiernya diikuti dengan tanda titik (.). perhatikan contoh berikut ini:

```

type
    struktur_mahasiswa = record
        nim : string[10];
        nama : string[25];
        alamat : string[20];
        nilai : longint;
    end;
var
    mahasiswa : struktur_mahasiswa;
begin
    mahasiswa.nim := '235121';
    mahasiswa.nama := 'Upin Ipin';
    mahasiswa.alamat:= 'Jl. Otista no 64c';
    mahasiswa.nilai:=98;
    writeln('NIM   : ',mahasiswa.nim);

```

```
writeln('Nama :',mahasiswa.nama);
writeln('Alamat  :',mahasiswa.alamat);
writeln('Nilai   :',mahasiswa.nilai);
readln;
end.
```

Statement `with` digunakan untuk menyingkat penulisan dalam membaca elemen record. Sintaks umumnya adalah `with namarecord do`. Potongan kode program di atas dapat kita ganti menjadi:

```
with mahasiswa do
begin
nim := '235121';
nama := 'Upin Ipin';
alamat:= 'Jl. Otista no 64c';
nilai:= 98;
end;
```

sehingga dengan menggunakan `with` maka kita tidak perlu untuk menuliskan nama record sebelum elemen recordnya.

Dalam contoh sebelumnya penggunaan tipe data record hanya dapat menyimpan satu data saja. (satu data mahasiswa saja) . Untuk dapat menyimpan sejumlah data mahasiswa maka dapat digunakan array yang bertipe record. Untuk mengakses tiap data mahasiswa pada array data mahasiswa, kita bisa memanfaatkan indeks array seperti pada pelajaran array sebelumnya. Perhatikan contoh berikut ini:

```
type
  data_mahasiswa = record
    nim : string[10];
    nama : string[25];
    alamat : string[20];
    nilai : longint;
  end;
var
  mahasiswa : array[1..10] of data_mahasiswa;
  i : integer;
begin
  for i:= 1 to 10 do
```



```

begin
    writeln('Input mahasiswa ke-', i);
with mahasiswa[i] do
begin
write('kode='); readln(nim);
write('nama='); readln(nama);
write('alamat='); readln(alamat);
write('nilai='); readln(nilai);
end;
end;

for i:=1 to 10 do
begin
writeln('Data Mahasiswa ke-', i);
write('kode='); writeln(mahasiswa[i].nim);
write('nama='); writeln(mahasiswa[i].nama);
write('alamat='); writeln(mahasiswa[i].alamat);
write('nilai='); writeln(mahasiswa[i].nilai);
end;
end.

```

Jika dalam suatu record terdapat beberapa field yang sama tipenya maka dapat kita gunakan array.

Contoh ada data barang yang mempunyai struktur:

- Nama barang -> bertipe String
- Jumlah unit barang ke 1 -> bertipe Byte
- Jumlah unit barang ke 2 -> bertipe Byte
- Jumlah unit barang ke 3 -> bertipe Byte

Maka deklarasi recordnya dapat kita nyatakan sebagai berikut:

```

type
    data_brg = record
        namaBrg : string[15];

```

```

    unitBrg : array[1..3] of byte;
end;
var
    Barang : array[1..10] of data_brg;

```

Untuk mengakses array unitBrg dalam record data\_brg maka dapat kita gunakan indeks array seperti contoh berikut:

```
Barang[1].unitBrg[1] := 5;
```

Berikut adalah contoh listing lengkap program:

```

type
    data_brg = record
        namaBrg : string[15];
        unitBrg : array[1..3] of byte;
    end;
var
    Barang : array[1..10] of data_brg;
    i : integer;
Begin
    Barang[1].namaBrg := 'Scanner';
    Barang[1].unitBrg[1] := 4;
    Barang[1].unitBrg[2] := 5;
    Barang[1].unitBrg[3] := 3;

    Barang[2].namaBrg := 'Printer';
    Barang[2].unitBrg[1] := 1;
    Barang[2].unitBrg[2] := 2;
    Barang[2].unitBrg[3] := 4;
    For i:=3 to 10 do begin
        With barang[i] do
            Begin
                Writeln('Barang ke-', i);
                Write('nama='); Readln(namaBrg);
                Write('jumlah 1= '); Readln(unitBrg[1]);
                Write('jumlah 2= '); Readln(unitBrg[2]);
                Write('jumlah 3= '); Readln(unitBrg[3]);
            End
        End
    End

```

```
End;  
End;  
End.
```

Tipe record dapat menjadi elemen dari record lainnya. Sebuah record dapat memiliki elemen yang juga bertipe record.

Contoh: sebuah data pegawai mempunyai struktur sebagai berikut :

- Nama pegawai -> string
- Mulai masuk -> - Tgl
  - Bln
  - Thn
- Alamat pegawai -> - Jalan
  - Kota
- Nilai -> - Nilai pokok
  - Lembur
  - Tunjangan

Maka deklarasi record data pegawai tersebut dapat kita nyatakan sebagai berikut:

```
type  
  masuk = record  
    tgl : 1..31;  
    bln : 1..12;  
    thn : integer;  
  end;  
  alamat = record  
    jalan : string[20];  
    kota : string[10];  
  end;  
  nilaipeg = record  
    pokok, tunjangan, lembur : real;  
  end;  
  pegawai = record  
    nama : string[20];  
    tglmasuk : masuk;
```

```

    almt : alamat;
    nilai : nilaipeg;
end;
var
    datapegawai : array [1..10] of pegawai;

```

atau bisa juga kita deklarasikan secara langsung sebagai berikut:

```

type
    pegawai = record
        nama: string[20];
        tglmasuk: record
            tgl : 1..31;
            bln : 1..12;
            thn : integer;
        end;
        alamat : record
            jalan : string[20];
            kota : string[10];
        end;
        nilaipeg : record
            pokok,tunjangan,lembur : real;
        end;
    end;
var
    datapegawai : array [1..100] of pegawai;

```

Untuk mengakses misalnya nilai pokok dari pegawai nomor 50. Maka caranya adalah seperti ini:

```
Datapegawai[50].nilaipeg.pokok := 5000000;
```

Untuk mengakses nama kota dari pegawai nomor 50 adalah sebagai berikut:

```
Datapegawai[50].alamat.kota := 'Bekasi';
```

### 11.5 Penugasan

Buatlah program entri data mahasiswa dengan memanfaatkan record dengan struktur nim, nama, kelas, dan nilai\_alpro. Program memiliki fungsi untuk input data, menampilkan data, menampilkan nilai terbesar, nilai terkecil, dan rata-rata. Semua fungsi memanfaatkan sub program yang dibahas pada pertemuan sebelumnya.

Selamat Datang di Program entri mahasiswa

1. Input Data
2. Tampilkan Data
3. Nilai Aplro Terbesar
4. Nilai Alpro Terkecil
5. Rata-rata Nilai Alpro
6. Keluar

## MODUL 12 REKURSI

---

### 12.1 Deskripsi Singkat

Rekursi adalah sub program yang memanggil dirinya sendiri baik secara langsung maupun tidak langsung. Pada modul 12 akan dibahas tentang rekursi, cara pembuatannya dan juga penggunaannya dalam program.

### 12.2 Tujuan Praktikum

Setelah praktikum pada modul 12 ini diharapkan:

1. mahasiswa memahami dan menguasai konsep rekursi
2. mahasiswa teknik memecahkan masalah pemrograman dengan menggunakan rekursi

### 12.3 Material Praktikum

Kegiatan pada modul 12 ini memerlukan material berupa program Pascal

### 12.4 Kegiatan Praktikum

Pada modul ini kita akan membahas tentang rekursi dimana pada bab sebelumnya kita telah membahas tentang sub program.

Ingat bahwa suatu fungsi atau prosedur dapat dipanggil dari fungsi atau prosedur lainnya.

Rekursi terjadi ketika sub program memanggil dirinya sendiri.

Rekursi banyak disandingkan dan dibandingkan dengan iterasi/perulangan untuk memudahkan pembelajaran dan membandingkan performanya.

#### A. Fungsi Rekursif

Berikut adalah contoh sebuah fungsi yang memanfaatkan iterasi untuk mengembalikan nilai faktorial dari sebuah bilangan.

```
Program faktorial_bilangan;
Uses crt;
Function Faktorial(x:integer):integer;
Var i, result:integer;
Begin
    Result:=1;
    For i:=x downto 1 do result:=result * i;
```

```

    Faktorial:=result;
End;
Var i: integer;
Begin
    Clrscr;
    i := Faktorial(5);
Writeln(i);
    Readln;
End.

```

Jalannya program dimulai dari baris pertama di program utama. Berikut adalah proses jalannya program:

1. Clrscr akan membersihkan layar dari kotoran
2. i:=Faktorial(5))

Baris ini akan meng-*assign* nilai dari Faktorial(5) ke variabel i. Berapakah nilai Faktorial(5)? Di sini fungsi Faktorial akan dipanggil dari program utama dengan argumen 5 yang akan ditransfer ke parameter x dalam fungsi Faktorial.

Maka for i:=x downto 1 pada fungsi tersebut akan menjadi for i:=5 downto 1. Perulangan result := result \* i akan diulang 5 kali sehingga hasilnya menjadi  $5*4*3*2*1 = 120$ .

Kembali ke program utama dengan mengembalikan nilai 120, i akan berisi 120.

3. Writeln(i) akan menuliskan 120 di layar, kemudian kursor turun 1 baris.
4. Readln akan membuat kursor berkedip sebelum program diakhiri.

Berikut ini adalah versi rekursi dari fungsi tersebut:

```

Function Faktorial(x:integer):integer;
Begin
    If x := 1 then faktorial := 1
Else Faktorial := x * Faktorial(x - 1);
End;

```

Berikut adalah jalannya fungsi ketika dipanggil Faktorial(5):

1. Faktorial(5) menjadi  $5 * \text{Faktorial}(4)$
2. Faktorial(4) menjadi  $4 * \text{Faktorial}(3)$
3. Faktorial(3) menjadi  $3 * \text{Faktorial}(2)$
4. Faktorial(2) menjadi  $2 * \text{Faktorial}(1)$
5. Faktorial(1) hasilnya adalah 1

6. Faktorial(2) hasilnya menjadi  $2 * 1 = 2$
7. Faktorial(3) hasilnya menjadi  $3 * 2 = 6$
8. Faktorial(4) hasilnya menjadi  $4 * 6 = 24$
9. Faktorial(5) hasilnya menjadi  $5 * 24 = 120$

Maka Faktorial(5) akan bernilai 120. Fungsi Faktorial dipanggil sebanyak 5 kali.

Contoh lainnya adalah sebuah fungsi dengan parameter  $x$  untuk mengembalikan nilai jumlah dari 1 sampai  $x$ . Berikut adalah versi iterasinya:

```
Function Sum(x:integer):integer;
Var i, result: integer;
Begin
    Result:=0;
    For i:=1 to x do result:=result+i;
    Sum:=result;
End;
```

Ketika misalnya dipanggil `sum(5)` maka jalannya fungsi adalah sebagai berikut:

1.  $x$  akan berisi 5;
2. Result bernilai awal 0;
3. For  $i:=1$  to 5 do  $result:=result+1$  akan menjumlahkan  $1+2+3+4+5 = 15$ .
4. Nilai 15 akan dikembalikan ke pemanggil fungsi tersebut.

Dan berikut ini adalah versi rekursinya

```
Function Sum(x: integer):integer;
Begin
    If x = 1 then Sum := 1
    Else Sum := x + Sum(x-1);
End;
```

Ketika misalnya dipanggil `sum(5)` maka jalannya fungsi adalah sebagai berikut:

1. Sum(5) menjadi  $5 + \text{sum}(4)$
2. Sum(4) menjadi  $4 + \text{sum}(3)$
3. Sum(3) menjadi  $3 + \text{sum}(2)$
4. Sum(2) menjadi  $2 + \text{sum}(1)$
5. Sum(1) hasilnya adalah 1
6. Sum(2) hasilnya menjadi  $2 + 1 = 3$



- 7. Sum(3) hasilnya menjadi  $3 + 3 = 6$
- 8. Sum(4) hasilnya menjadi  $4 + 6 = 10$
- 9. Sum(5) hasilnya menjadi  $5 + 10 = 15$

Fungsi Sum(5) akan menghasilkan 15.

## B. Prosedur Rekursif

Selain Function, Prosedur juga bisa bersifat rekursif, perhatikan contoh berikut ini:

```
PROCEDURE TULIS_1(banyak : integer; kata : string);  
begin  
if banyak > 1 then TULIS_1(banyak-1,kata);  
writeln(kata, banyak);  
end;
```

misal jika dipanggil TULIS\_1(5, 'Cetakan ke ') maka outputnya adalah:

```
Cetakan ke 1  
Cetakan ke 2  
Cetakan ke 3  
Cetakan ke 4  
Cetakan ke 5
```

Mengapa bisa demikian? Begini kira-kira jalannya prosedur:

1. Tulis\_1(5, 'Cetakan ke ') pada bagian if akan bernilai TRUE sehingga memanggil Tulis\_1(4, 'Cetakan ke ')
2. Tulis\_1(4, 'Cetakan ke ') pada bagian if akan bernilai TRUE sehingga memanggil Tulis\_1(3, 'Cetakan ke ')
3. Tulis\_1(3, 'Cetakan ke ') pada bagian if akan bernilai TRUE sehingga memanggil Tulis\_1(2, 'Cetakan ke ')
4. Tulis\_1(2, 'Cetakan ke ') pada bagian if akan bernilai TRUE sehingga memanggil Tulis\_1(1, 'Cetakan ke ')
5. Tulis\_1(1, 'Cetakan ke ') pada bagian if akan bernilai FALSE sehingga lanjut ke statemen selanjutnya yaitu writeln(kata, banyak) yang akan menuliskan Cetakan ke 1 di layar. Prosedur Tulis\_1(1, 'Cetakan ke ') selesai dijalankan dan kembali ke pemanggilnya yaitu Tulis\_1(2, 'Cetakan ke ')
6. Tulis\_1(2, 'Cetakan ke ') sudah selesai menjalankan bagian if, kemudian masuk ke writeln(kata, banyak) yang akan menuliskan Cetakan ke 2 di layar. Prosedur Tulis\_1(2, 'Cetakan ke ') selesai dijalankan dan kembali ke pemanggilnya yaitu Tulis\_1(3, 'Cetakan ke ')

7. Tulis\_1(3, 'Cetakan ke ') sudah selesai menjalankan bagian if, kemudian masuk ke writeln(kata, banyak) yang akan menuliskan Cetakan ke 3 di layar. Prosedur Tulis\_1(3, 'Cetakan ke ') selesai dijalankan dan kembali ke pemanggilnya yaitu Tulis\_1(4, 'Cetakan ke ')
8. Tulis\_1(4, 'Cetakan ke ') sudah selesai menjalankan bagian if, kemudian masuk ke writeln(kata, banyak) yang akan menuliskan Cetakan ke 4 di layar. Prosedur Tulis\_1(4, 'Cetakan ke ') selesai dijalankan dan kembali ke pemanggilnya yaitu Tulis\_1(5, 'Cetakan ke ')
9. Tulis\_1(5, 'Cetakan ke ') sudah selesai menjalankan bagian if, kemudian masuk ke writeln(kata, banyak) yang akan menuliskan Cetakan ke 5 di layar. Prosedur Tulis\_1(5, 'Cetakan ke ') selesai dijalankan.
10. Prosedur sudah selesai dijalankan dan menghasilkan output seperti di atas.

Bandingkan prosedur dan outputnya di atas dengan prosedur di bawah ini!

```
PROCEDURE TULIS_2(banyak : integer; kata : string);
begin
writeln(kata, banyak);
if banyak > 1 then TULIS_1(banyak-1,kata);
end;
```

OUTPUT (misal dipanggil dengan TULIS\_2(5, 'Cetakan ke '))

```
Cetakan ke 5
Cetakan ke 4
Cetakan ke 3
Cetakan ke 2
Cetakan ke 1
```

### C. Rekursi Tak Hingga

Sama dengan iterasi yang mungkin akan terjadi tak hingga bila kondisi pada while atau until tidak pernah terpenuhi, rekursi tak hingga juga dapat terjadi apabila jalannya pemanggilan sub program tidak pernah mencapai true pada kondisi if. Sebagai contoh, Apa yang terjadi bila dipanggil fungsi sum(0) pada kedua fungsi di bawah ini?

```
Function Sum(x:integer):integer;
Var i, result: integer;
Begin
```

```

Result:=0;
For i:=1 to x do result:=result+i;
Sum:=result;
End;

```

Untuk fungsi versi iterasi maka `sum(0)` hasilnya akan 0 (benar).

```

Function Sum(x: integer):integer;
Begin
  If x=1 then Sum:=1
  Else Sum:=x+Sum(x-1);
End;

```

Untuk fungsi versi rekursi `sum(0)` maka jalannya kira-kira seperti berikut:

- a. `Sum(0)` akan menjadi `0 + sum(-1)`
- b. `Sum(-1)` akan menjadi `-1 + sum(-2)`
- c. `Sum(-2)` akan menjadi `-2 + sum(-3)`
- d. `Sum(-3)` akan menjadi `-3 + sum(-4)`
- e. `Sum(-4)` akan menjadi `-4 + sum(-5)`
- f. `Sum(-5)` akan menjadi `-5 + sum(-6)`
- g. `Sum(-6)` akan menjadi `-6 + sum(-7)`
- h. `Sum(-7)` akan menjadi `-7 + sum(-8)`
- i. ....
- j. ....

Sampai kapan? Sampai selama-lamanya. Rekursi tak hingga akan terjadi jika kondisi pada `if` (disebut juga sebagai **base case** atau **special case**) tidak pernah terpenuhi.

#### D. Special Case pada Rekursi

Sub Program Rekursif akan memanggil dirinya sendiri selama kondisi pemanggilan dipenuhi. Dengan melihat sifat sub program rekursif tersebut maka sub program rekursif harus memiliki :

1. kondisi yang menyebabkan pemanggilan dirinya berhenti (disebut kondisi khusus atau special condition)
2. pemanggilan diri sub program (yaitu bila kondisi khusus tidak dipenuhi)

**Secara umum** bentuk dari sub program rekursif memiliki statemen kondisional :

*if kondisi khusus* tak dipenuhi

**then** *panggil diri-sendiri* dengan penyesuaian parameter

**else** *lakukan instruksi* yang akan dieksekusi bila kondisi khusus dipenuhi

**atau**

**if** *kondisi khusus* terpenuhi

**then** *lakukan instruksi* yang akan dieksekusi bila kondisi khusus dipenuhi

**else** *panggil diri-sendiri* dengan penyesuaian parameter

Sub program rekursif umumnya dipakai untuk permasalahan yang memiliki langkah penyelesaian yang terpola atau langkah-langkah yang teratur. Bila kita memiliki suatu permasalahan dan kita mengetahui algoritma penyelesaiannya, kadang-kadang sub program rekursif menjadi pilihan kita bila memang memungkinkan untuk dipergunakan.

Secara algoritmis (dari segi algoritma, yaitu bila kita mempertimbangkan penggunaan memori, waktu eksekusi sub program) sub program rekursif sering bersifat tidak efisien jika dibandingkan dengan sub program iteratif.

Dengan demikian sub program rekursif umumnya memiliki efisiensi dalam penulisan perintah, tetapi kadang tidak efisien secara algoritmis. Meskipun demikian banyak pula permasalahan-permasalahan yang lebih sesuai diselesaikan dengan cara rekursif (misalnya dalam pencarian/searching, yang mungkin akan dibahas pada pertemuan-pertemuan yang akan datang).

## **E. Merancang Algoritma Rekursi**

Dalam merancang suatu algoritma rekursif, strategi yang umum digunakan adalah “Divide and Conquer”. Terdapat beberapa pertanyaan yang harus dijawab seperti:

- Bagaimana kita dapat memecah suatu masalah menjadi beberapa masalah sama dalam versi yang lebih kecil.
- Bagaimana setiap pemanggilan fungsi dapat membuat masalah menjadi versi kecil.
- Apa *base case* dari masalah ini?
- Apakah algoritma akan selalu mencapai kondisi *base case*?

Untuk dapat membuat fungsi dan prosedur yang bersifat rekursif, berikut adalah langkah-langkahnya:

- Memahami masalah yang akan dipecahkan secara tepat. Langkah ini merupakan langkah awal dari seluruh permasalahan pemrograman.

- Tentukan seberapa besar masalah yang akan dipecahkan menjadi beberapa subprogram.
- Kenali dan tentukan base case dari masalah yang akan dikerjakan secara tidak rekursif.
- Terakhir kenali dan tentukan kondisi umum (general case) dengan benar

#### F. Fungsi Rekursif atau Prosedur Rekursif

Sebuah fungsi rekursif mungkin dapat kita ubah menjadi prosedur rekursif. Berikut ini adalah contoh fungsi rekursif untuk mengembalikan bilangan fibonacci suku ke n. **Barisan Bilangan Fibonacci** adalah **barisan** yang nilai sukunya sama dengan jumlah dua suku di depannya. **Barisan**: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

```
program fibo_using_rekursif;
var
  x,i: integer;
function fib(n:integer):integer;
begin
  if(n=1) then fib:=1
  else if (n=2) then fib:=1
  else fib:=fib(n-1)+fib(n-2);
end;
begin
  writeln('deret fibonacci');
  write('input value : ');
  readln(x);
  writeln;
  for i := 1 to x do write(fib(i), ' ');
  readln;
end.
```

Dan berikut ini adalah versi prosedurnya

```
procedure pib(n:integer; var hsl :integer);
var f1, f2: integer;
begin
  if(n=1) or (n=2) then hsl:=1
  else
    begin
```

```

        pib(n-1, f1);
        pib(n-2, f2);
        hsl:= f1 + f2;

    end;
end;
var x,i: integer; hsl:integer;
begin
    writeln('Barisan Bilangan Fibonacci');
    write('Jumlah bilangan Fibonnaci yang ingin
ditampilkan : ');
    readln(x); writeln;
    for i := 1 to x do
    begin
        pib(i,hsl);
        write(hsl, ' ');
    end;
    readln;
End.

```

## 12.5 Penugasan

- 1) Tambahkan fungsi faktorial pada program dengan kalkulator KATABAKU pada modul 8, yang dihitung dengan menggunakan rekursi.

Selamat datang di Kalkulator Sederhana  
Silahkan pilih menu berikut:

1. Penjumlahan
2. Pengurangan
3. Perkalian
4. Pembagian
5. Faktorial
6. Keluar

Pilihan Anda:

- 2) Buatlah program konversi nilai desimal ke biner dengan menggunakan rekursi

Input : 7  
Output :111

Input :10

```
Output :1010
```

## MODUL 13 PENCARIAN

---

### 13.1 Deskripsi Singkat

Istilah pencarian yang dibahas pada modul ini terbatas pada definisi menemukan suatu data dengan nilai tertentu pada sekumpulan data yang bertipe sama. Pada modul 13 ini akan dibahas tentang pencarian, algoritma pencarian, dan juga penggunaannya dalam program.

### 13.2 Tujuan Praktikum

Setelah praktikum pada modul 13 ini diharapkan:

1. mahasiswa memahami dan menguasai konsep pencarian sekuensial dan pencarian biner
2. mahasiswa menguasai teknik pencarian secara sekuensial dan biner

### 13.3 Material Praktikum

Kegiatan pada modul 13 ini memerlukan material berupa program Pascal

### 13.4 Kegiatan Praktikum

Pada modul ini kita akan membahas tentang pencarian. Proses pencarian adalah proses menemukan data tertentu di dalam sekumpulan data yang bertipe sama. Pada modul, sekumpulan data yang bertipe sama kita akan menggunakan array.

Pencarian adalah salah satu hal yang fundamental dalam pemrograman. Jika kita membuka aplikasi pada komputer, fitur yang hampir pasti selalu ada adalah fitur pencarian. Entah itu mencari sebuah kata dalam kumpulan teks, mencari data tertentu pada spreadsheet, mencari mahasiswa dengan nama tertentu pada aplikasi database dan lain sebagainya.

Pada modul praktikum ini akan dibahas 2 metode pencarian dasar yaitu pencarian sekuensial dan pencarian biner.

#### A. Pencarian Sekuensial pada Array Tidak Terurut

Pada pencarian sekuensial pada array yang tidak terurut, proses pencarian adalah proses iterasi yang dimulai dari data indeks pertama pada array sampai data yang dicari ditemukan atau sampai data pada indeks terakhir.

Dari pernyataan di atas, terdapat 2 kondisi dimana proses pencarian berhenti yaitu:



1. Jika data yang dicari ditemukan
2. Jika sudah sampai di data indeks terakhir.

Berikut adalah fungsi pencarian sekuensial untuk mencari data integer pada array of integer yang tidak terurut

```
Type Larik = array [1..100] of integer;
Function SequentialSearch(A:Larik; N:integer; X:integer)
: integer;
Var
    i:integer;
Begin
    i:=1;
    while (i<N) and (A[i] <> X) do i:= i + 1;
    if (A[i] <> X) then SequentialSearch:=0
    else SequentialSearch:=i;
End;
```

dimana i adalah indeks array, N adalah jumlah data, X adalah nilai yang mau dicari. Hasil dari fungsi di atas adalah nomor indeks array dimana data X ditemukan atau 0 jika tidak ada data yang dicari pada array.

## B. Pencarian Sekuensial pada Array Terurut

Pada pencarian sekuensial pada array yang terurut, proses pencarian adalah proses iterasi yang dimulai dari data indeks pertama pada array sampai data yang dicari ditemukan atau nilai yang dicari sudah melebihi atau kurang dari nilai data pada indeks array yang dikunjungi atau sampai data pada indeks terakhir.

Dari pernyataan di atas, terdapat 3 kondisi dimana proses pencarian berhenti yaitu:

1. Jika data yang dicari ditemukan
2. Jika nilai data pada indeks array yang sedang dikunjungi sudah melebihi atau kurang dari nilai yang dicari (tergantung array terurut menaik atau menurun).
3. Jika sudah sampai di data indeks terakhir.

Berikut adalah fungsi pencarian sekuensial untuk mencari data integer pada array of integer yang terurut dari kecil ke besar

```
Type Larik = array [1..100] of integer;
Function Sequentialsearch(A:Larik; N:integer; X:integer):
integer;
```

```

Var
    i:integer;
Begin
    i:=1;
    while (i<N) and (A[i] < X) do i:= i + 1;
    if (A[i] = X) then Sequentialsearch:=i
    else Sequentialsearch:=0;
End;

```

dimana i adalah indeks array, N adalah jumlah data, X adalah nilai yang mau dicari. Hasil dari fungsi di atas adalah nomor indeks array dimana data X ditemukan atau 0 jika tidak ada data yang dicari pada array.

### C. Pencarian Biner

Pencarian biner atau pencarian bagi dua hanya bisa dilakukan pada array yang terurut. Performa dari pencarian biner jauh lebih cepat dibandingkan pencarian sekuensial. Fungsi pencarian biner dapat dinyatakan sebagai fungsi rekursif atau iterative. berikut ini adalah fungsi pencarian biner secara iteratif

```

Function BinarySearch(A:larik; N: Integer; X:Integer):
integer;
var
    low, mid, high: integer;
begin
    low := 1;
    high := N;
    while (low <= high) do
    begin
        mid := (low + high) div 2;
        if (A[mid] > X) then high := mid - 1
        else if (A[mid] < X) then low := mid + 1
        else break;
    end;
    if A[mid] = X then BinarySearch:= mid {ditemukan}
    else BinarySearch := 0; {tidak ditemukan}
end;

```

dimana X adalah nilai yang hendak dicari, low adalah indeks array sebelah kiri, high adalah indeks array sebelah kanan, dan mid adalah indeks array tengah.

Berikut adalah contoh listing program lengkapnya:

```
program Pencarian;
Type Larik = array [1..100] of integer;

Function SequentialSearch(A:Larik; N:integer; X:integer)
: integer;
Var
    i:integer;
Begin
    i:=1;
    while (i<N) and (A[i] <> X) do i:= i + 1;
    if (A[i] <> X) then SequentialSearch:=0
    else SequentialSearch:=i;
End;

Function Sequentialsearch2(A:Larik; N:integer;
X:integer): integer;
Var
    i:integer;
Begin
    i:=1;
    while (i<N) and (A[i] < X) do i:= i + 1;
    if (A[i] = X) then Sequentialsearch2:=i
    else Sequentialsearch2:=0;
End;

Function BinarySearch(A:larik; N: Integer; X:Integer):
integer;
var
    low, mid, high: integer;
begin
    low := 1;
    high := N;
    while (low <= high) do
        begin
```

```

        mid := (low + high) div 2;
    if(A[mid] > X) then high := mid - 1
        else if(A[mid] < X) then low := mid + 1
            else break;
        end;
    if A[mid] = X then BinarySearch:= mid {ditemukan}
        else BinarySearch := 0; {tidak ditemukan}
end;
var
    A: larik;
    i,n,x : integer;
begin
    n:=10;
    x:=500;
    for i :=1 to n do A[i]:= i*100;
    writeln(SequentialSearch(A,n,x));
    writeln(SequentialSearch2(A,n,x));
    writeln(BinarySearch(A,n,x));
end.

```

### 13.5. Penugasan

- 1) Ubahlah fungsi pencarian biner di atas menjadi fungsi rekursi.
- 2) Melanjutkan program yang telah dikerjakan pada penugasan modul 11, tambahkan menu untuk mencari mahasiswa berdasarkan nama untuk kemudian menampilkan nilai alpronya

Selamat Datang di Program entri mahasiswa

1. Input Data
2. Tampilkan Data
3. Nilai Alpro Terbesar
4. Nilai Alpro Terkecil
5. Rata-rata Nilai Alpro
6. Cari Nilai Mahasiswa
7. Keluar

Pilihan Anda (1-7):

Jika pengguna memilih nomor 6 maka akan muncul permintaan nama mahasiswa:

6. Cari Nilai Mahasiswa

Masukkan nama mahasiswa: Panji

Data Ditemukan! Nilai Alpro Panji = 90

Atau jika data tidak ditemukan maka

6. Cari Nilai Mahasiswa

Masukkan nama mahasiswa: Andi

Data Tidak Ditemukan!

Menu pencarian dapat menggunakan pencarian sekuensial maupun biner.

## MODUL 14 PENGURUTAN

---

### 14.1 Deskripsi Singkat

Istilah pengurutan adalah proses menyusun kembali data dengan aturan tertentu. pada modul ini akan dibahas tentang pengurutan data pada sekumpulan data yang bertipe sama (array). Pada modul 14 ini akan dibahas tentang algoritma pengurutan data dasar, dan juga penggunaannya dalam program.

### 14.2 Tujuan Praktikum

Setelah praktikum pada modul 14 ini diharapkan:

1. mahasiswa memahami dan menguasai konsep pengurutan menggunakan bubble sort, selection sort, dan insertion sort
2. mahasiswa menguasai teknik pengurutan dan dapat menggunakannya dalam program

### 14.3 Material Praktikum

Kegiatan pada modul 14 ini memerlukan material berupa program Pascal

### 14.4 Kegiatan Praktikum

Pada modul ini kita akan membahas tentang pengurutan data menggunakan metode bubble, selection, dan insertion dengan pengurutan secara ascending.

#### A. Bubble Sort

Pada bubble sort, proses pengurutan dilakukan dengan cara membandingkan satu data dengan data berikutnya, jika lebih kecil maka akan ditukar urutannya.

Berikut adalah fungsi pengurutan dengan bubble sort

```
Type Larik = array [1..100] of integer;
Procedure BubbleSort(var data: Larik; n:integer);
var
    i,j,temp:integer;
Begin
    for i:=1 to n-1 do
        for j:=n downto i+1 do
            if (data[j]<data[j-1]) then begin
                temp := data[j];
                data[j] := data[j-1];
```

```

    data[j-1] := temp;
end;
End;

```

## B. Selection sort

Pada selection sort, proses pengurutan dilakukan dengan cara mencari nilai data terkecil atau terbesar pada setiap perulangan dan menempatkannya pada posisi yang sesuai. Selection sort merupakan kombinasi dari searching dan sorting. Berikut adalah prosedur untuk melakukan selection sort:

```

Type Larik = array [1..100] of integer;
Procedure SelectionSort(var Data:Larik; n:integer);
var
    i, j, min, temp:integer;
Begin
    For i:=1 to n-1 do
        begin
            Min:=i;
            For j:= i+1 to n do
                If Data[j] < Data[min] then Min:=j;
            Temp:=Data[i];
            Data[i]:=Data[min];
            Data[min]:=temp;
        End;
    End;
End;

```

## C. Insertion Sort

Pada insertion sort, proses pengurutan dilakukan seperti mengurutkan kartu pada satu tangan. Untuk menemukan posisi yang banar, maka satu persatu kartu yang ada di tangan harus dibandingkan secara berurutan. Berikut adalah prosedur untuk melakukan selection sort:

```

Type Larik = array [1..100] of integer;
Procedure InsertionSort(var data:larik; n:integer);
var
    i,j,key:integer;
begin
    for i:=2 to n do

```

```

begin
    key:=data[i];
    j:=i-1;
    while (j>0)and(data[j]>key) do begin
        data[j+1]:=data[j];
        j:=j-1;
    end;
    data[j+1]:=key;
end;
End;

```

Berikut adalah contoh listing program lengkapnya

```

program Urutin;
Type Larik = array [1..100] of integer;

Procedure BubbleSort(var data: Larik; n:integer);
var
    i,j,temp:integer;
Begin
    for i:=1 to n-1 do
        for j:=n downto i+1 do
            if (data[j]<data[j-1]) then begin
                temp := data[j];
                data[j] := data[j-1];
                data[j-1] := temp;
            end;
        end;
    end;
End;

Procedure SelectionSort(var Data:Larik; n:integer);
var
    i, j, min, temp:integer;
begin
    For i:=1 to n-1 do
        begin
            Min:=i;

```



```

    For j:= i+1 to n do
    begin
        If Data[j] < Data[min] then
            Min:=j;
    End;
    Temp:=Data[i];
    Data[i]:=Data[min];
    Data[min]:=temp;
    End;
End;

Procedure InsertionSort(var data:larik; n:integer);
var
    i,j,key:integer;
begin
    for i:=2 to n do
    begin
        key:=data[i];
        j:=i-1;
        while (j>0)and(data[j]>key) do begin
            data[j+1]:=data[j];
            j:=j-1;
        end;

        data[j+1]:=key;
    end;
End;
var
    data:larik;
    i,n:integer;
begin
    n:=5;
    data[1]:=5;
    data[2]:=8;
    data[3]:=6;

```

```

data[4]:=9;
data[5]:=1;
Insertionsort(data,n);
{Bubblesort(data,n);}
{Selectionsort(data,n);}
for i:=1 to n do writeln(data[i]);
end.

```

Secara performa, ketiga prosedur pengurutan tersebut memiliki kompleksitas yang sama atau setara. Tidak ada yang lebih unggul atau lebih cepat dari yang lainnya.

#### 14.5 Penugasan

- 1) Ubahlah ketiga prosedur pengurutan di atas menjadi pengurutan secara descending.
- 2) Melanjutkan program yang telah dikerjakan pada penugasan modul 13, tambahkan menu untuk mengurutkan mahasiswa berdasarkan nama dan juga mengurutkan mahasiswa berdasarkan nilai Alpro

```

Selamat Datang di Program entri mahasiswa
1. Input Data
2. Tampilkan Data
3. Nilai Alpro Terbesar
4. Nilai Alpro Terkecil
5. Rata-rata Nilai Alpro
6. Cari Nilai Mahasiswa
7. Urutkan Berdasarkan Nama
8. Urutkan Berdasarkan Nilai Alpro
9. Keluar
Pilihan Anda (1-9):

```

Jika pengguna memilih nomor 7 maka akan muncul pesan bahwa data telah diurutkan berdasarkan nama:

```

Pilihan Anda (1-9): 7
Data Telah Diurutkan Berdasarkan Nama!

```

Dan jika pengguna memilih nomor 8 maka akan muncul pesan bahwa data telah diurutkan berdasarkan nilai Alpro:

```

Pilihan Anda (1-9): 8
Data Telah Diurutkan Berdasarkan Nilai Alpro!

```

Pengurutan Nama sesuai dengan abjad dari kecil ke besar dan pengurutan nilai dilakukan secara descending dari besar ke kecil.

Untuk melihat apakah data telah terurut dengan benar maka anda bisa memilih menu nomor 2 untuk menampilkan data.