



# Pemrograman Dasar



Dhidhi Pambudi  
Dwi Maryono



Hak Cipta pada Kementerian Pendidikan dan Kebudayaan  
Dilindungi Undang-Undang

Milik Negara  
Tidak Diperdagangkan

Kontributor :  
Penyunting materi : Rosihan Ariyuana, S.Si., M.Kom  
Penyunting bahasa : Rosihan Ariyuana, S.Si., M.Kom  
Desain Sampul : Punto Wicaksono, Bayu Adi Prasetyo Wibowo

Katalog dalam terbitan

Cetakan ke 1, 2014  
Disusun dengan huruf Arial 11pt

## KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan yang Maha Esa atas tersusunnya buku teks ini, dengan harapan dapat digunakan sebagai buku teks Pemrograman Dasar untuk siswa Sekolah Menengah Kejuruan (SMK) Bidang Keahlian Teknologi Informasi.

Penerapan kurikulum 2013 mengacu pada paradigma belajar kurikulum abad 21 menyebabkan terjadinya perubahan, yakni dari pengajaran (*teaching*) menjadi BELAJAR (*learning*), dari pembelajaran yang berpusat kepada guru (*teachers-centered*) menjadi pembelajaran yang berpusat kepada peserta didik (*student-centered*), dari pembelajaran pasif (*pasive learning*) ke cara belajar peserta didik aktif (*active learning-CBSA*) atau *Student Active Learning-SAL*.

Buku teks "Pemrograman Dasar" ini disusun berdasarkan tuntutan paradigma pengajaran dan pembelajaran kurikulum 2013 diselaraskan berdasarkan pendekatan model pembelajaran yang sesuai dengan kebutuhan belajar kurikulum abad 21, yaitu pendekatan model pembelajaran berbasis peningkatan keterampilan proses sains.

Penyajian buku teks untuk Mata Pelajaran Pemrograman Komputer ini disusun dengan tujuan agar peserta didik dapat melakukan proses pencarian pengetahuan berkenaan dengan materi pelajaran melalui berbagai aktivitas proses sains sebagaimana dilakukan oleh para ilmuwan dalam melakukan eksperimen ilmiah (penerapan pendekatan saintifik), dengan demikian peserta didik diarahkan untuk menemukan sendiri berbagai fakta, membangun konsep, dan nilai-nilai baru secara mandiri. Pendekatan saintifik ini meliputi proses 5 M,yaitu mengamati, menanya, mencoba/mengumpulkan informasi, mengassosiasikan, dan mengkomunikasikan.

Kementerian Pendidikan dan Kebudayaan, Direktorat Pembinaan Sekolah Menengah Kejuruan, dan Direktorat Jenderal Peningkatan Mutu Pendidik dan Tenaga Kependidikan menyampaikan terima kasih, sekaligus saran kritik demi kesempurnaan buku teks ini dan penghargaan kepada semua pihak yang telah berperan serta dalam membantu terselesaikannya buku teks siswa untuk Mata Pelajaran "Pemrograman Komputer Kelas XI Semester 1 Sekolah Menengah Kejuruan (SMK).

Jakarta, Desember 2014

Menteri Pendidikan dan Kebudayaan

## **DAFTAR ISI**

Kata Pengantar .....	ii
Daftar Isi .....	iii
Pendahuluan .....	iv
Bab I. Prosedur dan Fungsi .....	1
KB 1. Definisi dan Deklarasi Prosedur .....	2
KB 2. Pemanfaatan Prosedur .....	15
KB 3. Definisi dan Deklarasi Fungsi .....	27
KB 4. Pemanfaatan Prosedur dan Fungsi dalam Aplikasi .....	39
KB 5. Fungsi Rekursif dan Aplikasinya .....	51
KB 6. Pointer ke Fungsi dan Aplikasinya .....	63
KB 7. Modularisasi Fungsi .....	69
Bab II. Pencarian dan Pengurutan Data .....	83
KB 1. Pencarian Data dengan Algoritma Linear .....	84
KB 2. Pemanfaatan Pencarian Data dalam Aplikasi .....	93
KB 3. Pengurutan Data dengan Algoritma <i>Bubble Sort</i> .....	97
KB 4. Pengurutan Data dengan Algoritma <i>Selection Sort</i> .....	113
KB 5. Pemanfaatan Pengurutan Data dalam Aplikasi .....	128
Bab III. Pengembangan Aplikasi .....	137
KB 1. Model Waterfall Tahap Analisis .....	138
KB 2. Model Waterfall Tahap Desain .....	147
KB 3. Model Waterfall Tahap Pengujian .....	151
KB 4. Model Prototyping .....	155
Daftar Pustaka .....	159

## PENDAHULUAN

Buku Pemrograman Dasar ini ditujukan untuk SMK/MAK bidang keahlian Teknologi Informasi dan Komunikasi kelas XI semester 2. Buku ini dirancang untuk menjadi buku pegangan siswa agar bisa digunakan untuk membantu pembelajaran yang menerapkan Kurikulum 2013 (K13). Buku ini bukanlah pedoman utama dalam pembelajaran berbasis Kurikulum 2013 tetapi buku ini hanya sebagai salah satu sumber belajar yang mengarah pada penerapan Kurikulum 2013. Keberhasilan proses pembelajaran bukan ditentukan semata-mata oleh buku ini tetapi lebih ditekankan pada bagaimana siswa bisa lebih berkembang dengan menerapkan cara belajar dan berpikir *scientific*.

Buku ini sudah dirancang sedemikian rupa agar mudah dipelajari. Supaya siswa lebih mudah menguasai setiap materi yang disajikan maka siswa diharapkan sudah memenuhi standar ketuntasan belajar dalam pelajaran Pemrograman Dasar kelas XI semester 1. Siswa diharapkan banyak mencoba dan mengembangkan kreativitasnya dalam pembuatan program. Buku ini menggunakan bahasa pemrograman Pascal dan *compiler* yang digunakan dalam penyusunan buku adalah Free Pascal. Diharapkan siswa sudah terbiasa menggunakan *compiler* dan *Integrated Development Environment* (IDE) Free Pascal.

Penguasaan materi buku ini bisa dilakukan dengan cara mempelajarinya secara terurut mulai dari Bab I sampai Bab III dan mempelajari setiap Kegiatan Belajar (KB) secara terurut pula. Pada setiap Kegiatan Belajar siswa diminta untuk benar-benar membaca isi pada bagian kegiatan Mengamati dan selanjutnya siswa diharapkan memunculkan pertanyaan dari dirinya sendiri, meskipun dalam buku ini sudah disediakan beberapa pertanyaan yang berkaitan dengan materi. Setiap pertanyaan yang sudah disediakan dalam buku ini bisa dijawab oleh siswa setelah melakukan beberapa percobaan yang sudah disediakan pada kegiatan selanjutnya. Meskipun tidak selalu, tetapi pada umumnya setiap nomor pertanyaan akan bersesuaian dengan nomor eksperimen, yaitu: pertanyaan nomor satu bersesuaian dengan eksperimen nomor satu, pertanyaan nomor dua bersesuaian dengan eksperimen nomor dua, dan seterusnya.

**Kompetensi Inti:**

- K1. Menghayati dan mengamalkan ajaran agama yang dianutnya
- K2. Menghayati dan mengamalkan perilaku jujur, disiplin, tanggungjawab, peduli (gotong royong, kerjasama, toleran, damai), santun, responsif dan pro-aktif dan menunjukkan sikap sebagai bagian dari solusi atas berbagai permasalahan dalam berinteraksi secara efektif dengan lingkungan sosial dan alam serta dalam menempatkan diri sebagai cerminan bangsa dalam pergaulan dunia.
- K3. Memahami, menerapkan, dan menganalisis pengetahuan faktual, konseptual, prosedural, dan metakognitif berdasarkan rasa ingin tahu tentang ilmu pengetahuan, teknologi, seni, budaya, dan humaniora dalam wawasan kemanusiaan, kebangsaan, kenegaraan, dan peradaban terkait penyebab fenomena dan kejadian dalam bidang kerja yang spesifik untuk memecahkan masalah.
- K4. Mengolah, menalar, dan menyaji dalam ranah konkret dan ranah abstrak terkait dengan pengembangan dari yang dipelajarinya di sekolah secara mandiri, bertindak secara efektif dan kreatif, dan mampu melaksanakan tugas spesifik di bawah pengawasan langsung.

**Kompetensi Dasar:**

- 1.1. Memahami nilai-nilai keimanan dengan menyadari hubungan keteraturan dan kompleksitas alam dan jagad raya terhadap kebesaran Tuhan yang menciptakannya
- 1.2. Mendeskripsikan kebesaran Tuhan yang menciptakan berbagai sumber energi di alam
- 1.3. Mengamalkan nilai-nilai keimanan sesuai dengan ajaran agama dalam kehidupan sehari-hari
- 1.4. Menunjukkan perilaku ilmiah (memiliki rasa ingin tahu; objektif; jujur; teliti; cermat; tekun; hati-hati; bertanggung jawab; terbuka; kritis; kreatif; inovatif dan peduli lingkungan) dalam aktivitas sehari-hari sebagai wujud implementasi sikap dalam melakukan percobaan dan berdiskusi
- 1.5. Menghargai kerja individu dan kelompok dalam aktivitas sehari-hari sebagai wujud implementasi melaksanakan percobaan dan melaporkan hasil percobaan.

## **PEMBELAJARAN**

Materi:

- Bab I. Prosedur dan Fungsi
- Bab II. Pencarian dan Pengurutan Data
- Bab III. Pengembangan Aplikasi

# BAB

## I

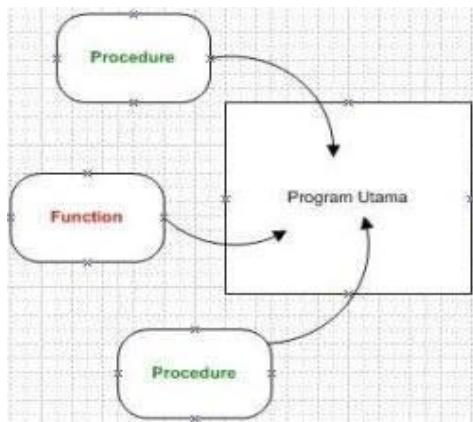
### ***PROSEDUR DAN FUNGSI***

#### Kompetensi Dasar:

- Mendeskripsikan penggunaan fungsi
- Memahami penggunaan fungsi rekursif
- Menerapkan pointer ke fungsi
- Menerapkan header file dalam pemrograman
- Memecahkan masalah prosedural menggunakan konsep fungsi
- Memecahkan masalah prosedural menggunakan konsep fungsi rekursif
- Memodifikasi data melalui pointer ke fungsi
- Memodifikasi program prosedural menggunakan header file

# BAB I

## PROSEDUR DAN FUNGSI



Ketika program yang dibuat sudah terlalu panjang ratusan bahkan puluhan ribu baris, sehingga kita mengalami kesulitan untuk memahami jalannya program secara keseluruhan, maka ada baiknya bila program tersebut dipecah menjadi beberapa bagian yang biasanya disebut modul, subprogram atau subrutin. Konsep semacam ini biasa disebut dengan pemrograman prosedural. Dalam tulisan ini selanjutnya akan digunakan kata **subprogram** supaya lebih ringkas. Memecah program menjadi subprogram-subprogram tentunya akan lebih memudahkan dalam mencari kesalahan, memperbaiki serta membuat dokumentasinya. Pembuatan subprogram di FreePascal dibagi dua jenis yaitu : Prosedur dan Fungsi. Prosedur atau Fungsi adalah suatu program yang terpisah dari program utama, diletakan dalam blok tersendiri yang berfungsi sebagai bagian dari program. Setiap prosedur diawali dengan kata tercadang (*reserved word*) **Procedure**, sedangkan fungsi diawali dengan kata tercadang **Function**.

### 1.1. Kegiatan Belajar 1. Deklarasi dan Definisi Prosedur

**Alokasi Waktu : 2 x 45 menit**

#### 1.1.1. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 1, siswa diharapkan dapat :

1. Menjelaskan definisi procedure
2. Mendeklarasikan procedure
3. Menggunakan parameter dalam procedure
4. Menyelesaikan masalah menggunakan procedure

#### 1.1.2. Aktivitas belajar siswa

##### 1.1.2.1. Mengamati/ observasi

Prosedur diawali dengan kata tercadang **Procedure** di dalam bagian deklarasi prosedur. Prosedur dipanggil dan digunakan di dalam blok program yang lainnya dengan menyebutkan judul prosedurnya.

Sebagaimana halnya sebuah program, suatu *procedure* juga memiliki *header* dan blok. Perbedaan bentuknya dengan program hanyalah pada bagian *header*-nya saja.

Bentuk umum *header* suatu *procedure* adalah :

```
procedure nama;
```

atau

```
procedure nama (parameter_formal : tipe_data);
```

Jika kita menggunakan *procedure* dalam suatu program, maka *procedure* tersebut harus dituliskan pada bagian deklarasi.

### Jangkauan Variabel

Variabel yang dideklarasikan dalam suatu blok program hanya berlaku pada blok dimana variabel tersebut didefinisikan. Perhatikan contoh program berikut.

```
program p;
var x : real;

procedure pl;
var y : integer;
begin
    ....;
    ....;
end;

// Program Utama
begin
    ....;
    ....;
    ....;
end.
```

**Keterangan :** daerah berlakunya variabel x mencakup semua perintah dalam program tersebut, namun untuk variabel y hanya berlaku di bagian subprogram *procedure* saja.

## Procedure dengan Parameter

Nilai dalam suatu subprogram FreePascal sifatnya adalah lokal, artinya hanya dapat digunakan pada modul atau unit yang bersangkutan saja, tidak dapat digunakan pada modul atau unit program yang lainnya. Parameter adalah data masukan untuk subprogram yang nantinya akan diproses lebih lanjut dalam subprogram tersebut. Dalam Pascal, dikenal dua macam parameter, yaitu :

1. parameter nilai (*value parameter*), dan
2. parameter referensi (*reference parameter*).

Cara mendeklarasikan parameter tersebut adalah sebagai berikut :

```
procedure B(namaParam : tipeData; var namaParam :  
           tipeData);  
begin  
    { statement }  
end;
```

Untuk menggunakan sebuah prosedur dalam program utama adalah cukup dengan memanggil nama prosedur dan memberikan parameter yang sesuai jika prosedur yang dipanggil membutuhkan parameter. Contoh :

```
procedure tanya_hitung;  
var X,Y : real;  
begin  
    write ('Nilai X ?');  
    readln (X);  
    Y:=X*X;  
    writeln ('Nilai Y = ', Y : 6:2);  
end;  
  
// Program Utama  
begin  
    Tanya_hitung;  
end.
```

Program ini jika di *compile* dan di *run*, hasilnya adalah :

```
Nilai X ? 5  
Nilai Y = 25.00
```

**Keterangan :**

Variabel X dan Y sifatnya adalah lokal untuk prosedur tanya\_hitung, artinya hanya dapat digunakan pada modul itu saja. Pada modul yang lain tidak dapat digunakan.

### **Pengiriman Parameter (*passing parameter*)**

Proses pengiriman data dari parameter aktual ke parameter formal disebut dengan transfer parameter (*passing parameter*). Nama parameter aktual dan parameter formal boleh berbeda, tetapi harus memiliki tipe data yang sama selain itu juga jumlah parameter aktual dan parameter formal harus sama. Di FreePascal, parameter dapat dikirim secara nilai (*by value*) atau secara acuan (*by reference*).

a) Pengiriman parameter secara nilai (*by value*)

Jika parameter dikirim secara nilai, maka parameter formal yang terletak di dalam pendefinian prosedur akan berisi nilai yang dikirimkan dari parameter aktual, yang kemudian bersifat lokal di prosedur tersebut. Bila nilai parameter formal di dalam prosedur tersebut berubah, maka tidak akan mempengaruhi nilai parameter aktual (nilai parameter aktual tetap). Pengiriman parameter secara nilai biasanya terjadi pada jenis parameter masukan (*input parameter*).

b) Pengiriman parameter secara acuan (*by reference*)

Bila parameter dikirimkan secara acuan, maka perubahan-perubahan yang terjadi pada nilai parameter formal di prosedur akan mempengaruhi nilai parameter aktual di blok program utama. Jenis parameter ini dideklarasikan di dalam prosedur dengan menggunakan kata cadang *var*.

### **Acuan Forward**

Digunakan untuk mendeklarasikan dimuka judul prosedur terpisah dari bloknya. Kegunaan utama dari mekanisme forward ini adalah untuk teknik rekursif dengan menggunakan dua prosedur yang saling memanggil.

Contoh :

```
Var I : integer;
```

```
Procedure pro1(var I : integer); Forward;

Procedure pro2(var I : integer);
Begin
    Writeln('prosedur pro',I);
End;

Procedure pro1;
Begin
    Writeln('prosedur pro',I);
End;

// Program Utama
Begin
    I := 1; pro1(I);
    I := 2; pro2(I);
End.
```

Hasilnya :

```
Prosedur pro1
Prosedur pro2
```

Prosedur tercadang yang disediakan oleh FreePascal antara lain :

1. Prosedur standar EXIT

Digunakan untuk keluar dari suatu blok.

2. Prosedur standar HALT

Digunakan untuk menghentikan program baik di program bagian maupun di program utama.

3. Prosedur standar MOVE

Bentuk umum : MOVE (var source, dest; count : word);

Digunakan untuk menyalin suatu blok sebanyak count byte memori dari blok dimulai byte pertama source dan disalinkan ke byte pertama dest.

4. Prosedur standar FILLCHAR

Digunakan untuk mengisi sejumlah byte bilai ke dalam suatu variabel, sebagai berikut :

FillChar (x; count : word; ch);

X adalah variabel yang dapat bertipe apapun yang akan diisi dengan nilai tipe ordinal Ch sebanyak count byte.

Perhatikan pernyataan berikut!

Misalkan terdapat tiga bilangan berbeda. Bagaimana cara membuat program yang bisa menentukan bilangan bulat terbesar diantara tiga bilangan bulat tersebut? Untuk lebih jelasnya, perhatikan program untuk mengetahui nilai terbesar dari tiga bilangan a,b,c dengan bantuan subprogram *procedure*.

```
procedure maksimum;
var max : integer;
begin
    if a > b then max := a else max := b;
    if c > max then max := c;
    writeln (max);
end;
```

Program tersebut hanya merupakan deklarasi *procedure*, untuk kemudian dapat digunakan dengan bentuk tulisan sebagai berikut :

```
program contoh_1;
var a,b,c : integer;

procedure maksimum;
var max : integer;
begin
    if a > b then max := a else max := b;
    if c > max then max := c;
    writeln (max);
end;

//program utama
begin
    readln (a, b, c);
    maksimum;
end.
```

### 1.1.2.2. Menanya

Berdasarkan kegiatan mengamati, ada beberapa hal yang bisa diperhatikan dan dibahas lebih lanjut. Beberapa pertanyaan yang berkaitan dengan prosedur adalah:

1. Bagaimana cara memodifikasi program sehingga memiliki prosedur?
2. Bagaimana cara membuat program yang menggunakan lebih dari satu procedure?

Untuk menjawab pertanyaan-pertanyaan tersebut, siswa diharapkan bisa melakukan kegiatan mencoba berikut dengan bimbingan guru.

### 1.1.2.3. Mencoba

#### Percobaan 1

Di bawah ini adalah program untuk mencari luas dan keliling persegi panjang. Dalam bab 1 semester ketiga, sudah dipelajari tentang penggunaan operator dan fungsi-fungsi aritmatika dalam memecahkan permasalahan dengan cara biasa.

```
program PersegiPanjang;                      {Judul Program Utama}
uses crt;
var
    p,l:byte;                                {deklarasi variabel global}
    ls,kel:integer;                          {p :panjang, l:lebar}
                                            {ls:luas, kel:keliling}
begin
    clrscr;
    writeln('Luas dan Keliling persegi panjang');
    writeln;
    write('Masukan panjang : ');readln(p);
    write('Masukan lebar   : ');readln(l);
    writeln;
    ls:=p*l;                                {luas=panjang x lebar}
    Kel:=(2*p)+(2*l);                      {keliling=(2 x panjang)+(2 x lebar) }
    writeln('Luas      : ',ls);
    writeln('Keliling : ',kel);
    readln;
end.
```

Lalu, bagaimana jika dimodifikasi agar program tersebut menggunakan procedure?

Cara paling sederhana adalah memindah proses yang adalam program utama menjadi sebuah prosedur. Salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```
program PersegiPanjang;
uses crt;
var p,l:byte;                                {p :panjang, l:lebar}
    ls,kel:integer;                           {ls:luas, kel:keliling}

procedure .......;
begin
    write('Masukan panjang : ');readln(p);
    write('Masukan lebar   : ');readln(l);
    writeln;
    ls:=.....;                               {luas=panjang x lebar}
    .....:=(2*p)+(2*l);           {keliling=(2 x panjang)+(2 x lebar) }
    writeln('Luas      : ',.....);
    writeln('Keliling : ',kel);
end;

//program utama
begin
    clrscr;
    writeln('Luas dan Keliling persegi panjang');
    writeln;

    luas; //pemanggilan prosedur

    readln;
end.
```

Cara yang lain misalnya hendak dibuat prosedur dengan parameter, dimana parameter yang dipilih adalah panjang dan lebar. Salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```

program PersegiPanjang;           {Judul Program Utama}
uses crt;
var
  p,l:byte;                      {deklarasi variabel global}
  ls,kel:integer;                {p :panjang, l:lebar}
                                {ls:luas, kel:keliling}

procedure Luas(pj, lb : byte);
begin
  ls:=pj*lb;                     {luas=panjang x lebar}
  .....:=(2*pj)+(2*....);        {
  writeln('Luas      : ',ls);
  writeln('Keliling : ',kel);
end;

//program utama
begin
  clrscr;
  writeln('Luas dan Keliling persegi panjang');
  writeln;
  write('Masukan panjang : ');readln(p);
  write('Masukan lebar   : ');readln(l);
  writeln;

  Luas(p,l); //pemanggilan prosedur dengan parameter yang sesuai
  writeln;
end.

```

Cara yang lain lagi, dengan memperhatikan bahwa variabel ls dan kel adalah variabel global maka bisa saja dibuat prosedur seperti berikut.

```

program PersegiPanjang;           {Judul Program Utama}
uses crt;
var p,l:byte;                    {p :panjang, l:lebar}
  ls,kel:integer;                {ls:luas, kel:keliling}

procedure Luas(pj, lb : byte);
begin
  ls:=pj*lb;                     {luas=panjang x lebar}
  .....:=(2*pj)+(2*....);
end;

```

```
//program utama
begin
  clrscr;
  writeln('Luas dan Keliling persegi panjang');
  writeln;
  write('Masukan panjang : ');readln(p);
  write('Masukan lebar    : ');readln(l);
  writeln;

  Luas(p,l); //pemanggilan prosedur dengan parameter yang sesuai

  writeln('Luas      : ',ls);
  writeln('Keliling : ',kel);
  readln;
end.
```

Modifikasi yang lain bisa dibuat prosedur yang menggunakan parameter acuan sebagai output dari prosedur, misalnya keliling dijadikan parameter acuan untuk output prosedur. Salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```
program PersegiPanjang;           {Judul Program Utama}
uses crt;
var p,l:byte;                    {p :panjang, l:lebar}
   ls,kel:integer;                {ls:luas, kel:keliling}

procedure Luas(pj, lb : byte; var keliling:integer);
begin
  ls:=pj*lb;                     {luas=panjang x lebar}
  .....:= (2*pj)+(2*....);       {
  writeln('Luas      : ',ls);
end;

//program utama
begin
  clrscr;
  writeln('Luas dan Keliling persegi panjang');
  writeln;
  write('Masukan panjang : ');readln(p);
```

```
write('Masukan lebar : ');readln(l);
writeln;

Luas(p,l,kel); //pemanggilan prosedur dengan parameter yang
sesuai
writeln('Keliling : ',kel);

readln;
end.
```

## Percobaan 2

Salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```
program BanyakProsedur;
uses wincrt;
var
c,d:integer;
j:real;
procedure jumlah(a,b:integer; jum:real);
begin
    jum:=a+b;
    writeln('Jumlah A+B      =', jum:6:2);
end;
procedure pembagi(a,b:integer; bagi:real);
begin
    bagi:=a/b;
    writeln('Bagi A/B      =', bagi:6:2);
end;
procedure pengurang(a,b:integer; kurang:real);
begin
    kurang:=a-b;
    writeln('Kurang A-B      =', kurang:6:2);
end;
procedure perkalian(a,b:integer; kali:real);
begin
    kali:=a*b;
```

```

        writeln('Kali A*B      =',kali:6:2);
    end;
begin
    write('Masukan Nilai A  : ');readln(c);
    write('Masukan Nilai B  : ');readln(d);
    jumlah(c,d,j);
    pembagi(c,d,j);
    pengurang(c,d,j);
    perkalian(c,d,j);
end.

```

#### Hasil Percobaan Program Banyak Prosedur:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

#### 1.1.2.4 Mengasosiasi/ menalar

Setelah siswa melakukan percobaan yang sesuai dengan pertanyaan yang ada, siswa diharapkan melakukan kegiatan menalar yang bisa dilakukan secara individu oleh masing-masing siswa atau bisa juga berkelompok. Beberapa hal yang bisa dijadikan arahan untuk menalar :

1. Perhatikan hasil kompilasi, apakah berhasil atau justru terjadi kesalahan.
2. Perhatikan urutan proses dari setiap program yang sudah dicoba.
3. Coba bandingkan beberapa program yang memiliki kemiripan, program mana yang berhasil dikompilasi dan program mana yang tidak berhasil dikompilasi.
4. Perhatikan letak/posisi kesalahan, coba pikirkan mengapa terjadi kesalahan pada posisi tersebut.
5. Cobalah membuat kesimpulan dari percobaan yang telah dilakukan.

#### 1.1.3. Rangkuman

Dari percobaan-percobaan yang telah dilakukan, bisa diambil kesimpulan bahwa:

1. Prosedur atau Fungsi adalah suatu program yang terpisah dari program utama, diletakan dalam blok tersendiri yang berfungsi sebagai bagian dari program.

2. Suatu program memerlukan subprogram untuk mempermudah dalam pembuatannya serta subprogram digunakan untuk mempersingkat penulisan, procedure dalam bahasa Pascal termasuk ke dalam subprogram. Penulisan program dengan menggunakan procedure menunjukkan teknik pemrograman yang baik dan terstruktur.
3. Dalam bahasa Pascal, pendefinisian prosedur ditulis bersatu di dalam program utama, kecuali jika direalisasikan sebagai unit. Prosedur diletakkan di bawah kata `var`.
4. Prosedur dapat menggunakan parameter atau tanpa menggunakan parameter. Parameter berfungsi sebagai media komunikasi antara subprogram dengan program pemanggil. Selain itu, parameter dapat mengurangi kebutuhan penggunaan peubah global.
5. Bahasa Pascal memungkinkan prosedur mempunyai parameter masukan, parameter keluaran, dan parameter masukan/keluaran. Parameter formal yang bertipe keluaran atau masukan/keluaran harus diawali dengan kata kunci `var`, sedangkan parameter formal yang bertipe masukan tidak diawali dengan kata kunci `var`. Argumen parameter aktual dilewatkan ke parameter formal yang bertipe masukan sebagai “*by value*”, sedangkan bila parameter formalnya bertipe masukan atau masukan/keluaran, maka argumen parameter aktual dilewatkan sebagai “*by reference*”.
6. Untuk mengetahui apakah suatu parameter termasuk parameter masukan atau parameter keluaran adalah dengan mengetahui masalahnya, apakah prosedur tersebut menghasilkan keluaran yang digunakan oleh program pemanggil atau tidak. Bila prosedur menghasilkan keluaran yang digunakan oleh program pemanggil, gunakanlah parameter keluaran untuk menampung keluaran tersebut. Sebaliknya, bila prosedur tidak menghasilkan keluaran, ataupun kalau menghasilkan keluaran dan ternyata keluaran tersebut hanya digunakan di dalam prosedur itu saja, gunakan parameter masukan. Bila prosedur menerima masukan sekaligus keluaran pada parameter yang sama, gunakan parameter masukan /keluaran.
7. Procedure dapat digunakan juga untuk tipe data yang rumit, dengan catatan keluaran hanya data tunggal.
8. Parameter masukan merupakan parameter yang nilainya berfungsi sebagai nilai awal proses pada procedure. Nilai parameter aktual yang disertakan ketika memanggil nama procedure menjadi nilai masukan pada parameter formal sebuah procedure.
9. Pada procedure tanpa parameter dan dengan parameter masukan hanya mengerjakan proses yang sebenarnya merupakan masukan program utama yang dikerjakan pada subprogram. Output langsung ditampilkan dalam procedure tersebut. Beda halnya dengan procedure dengan parameter keluaran. Nilai data parameter masukan (parameter aktual) yang dimasukkan pada parameter formal akan diolah dan diproses kemudian menghasilkan nilai keluaran melalui parameter formal yang sudah dideklarasikan pada *header procedure*.
10. Pada struktur *procedure* dengan parameter dan keluaran, setiap parameter aktual yang dikirimkan ketika memanggil procedure akan digunakan sebagai parameter formal yang

- menjadi nilai awal proses dan kemudian akan dikembalikan melalui parameter formal pada program utama.

### **1.1.2. Tugas**

Setelah mengikuti kegiatan belajar di atas, berikutnya siswa bisa memperdalam pengetahuannya dan berlatih membuat program sendiri untuk mengasah kemampuan pembuatan program yang memanfaatkan prosedur.

- Deklarasi prosedur manakah yang dibenarkan?
  - procedure hapus;
  - procedure hapus(s:string);
  - procedure hapus(var s:string);
  - procedure hapus(s:string):boolean;
  - procedure hapus(var data);
- Buatlah prosedur yang menuliskan bintang (\*) sebanyak n kali, dimana n adalah parameter masukannya.
- Buatlah prosedur untuk menghitung gaya yang dihasilkan berdasar massa dan percepatan yang diketahui. Dengan massa dan percepatan bertipe integer.
- Buatlah program yang membaca dua buah bilangan bertipe integer yang disimpan pada dua buah variabel dan menukar nilai variabel tersebut.
- Buatlah program yang menerima 3 buah bilangan bertipe integer (angka1,angka2,angka3) dan menggeser nilai-nilai yang disimpan oleh variabel tersebut sehingga isi angka1 menjadi isi angka2, isi angka2 menjadi isi angka3, dan isi angka3 menjadi isi angka1.

## **1.2. Kegiatan Belajar 2. Pemanfaatan Prosedur**

**Alokasi Waktu : 2 x 45 menit**

### **1.2.1. Tujuan Pembelajaran**

Setelah mengikuti kegiatan belajar 2, siswa diharapkan dapat :

- Menggunakan definisi procedure dalam penyelesaian masalah sehari-hari, seperti aplikasi bisnis, palindrom atau permasalahan lainnya.

### **1.2.2. Aktivitas belajar siswa**

#### **1.2.2.1. Mengamati/ observasi**

Kalian pasti pernah menemui atau menggunakan aplikasi konversi waktu bukan? Suatu aplikasi yang dibuat untuk menentukan berapa detik yang dihasilkan dari menit dan jam yang dimasukkan oleh pengguna. Dalam bahasa pemrograman, aplikasi konversi biasa dibuat untuk

mengenalkan pembuat program pemula akan logika berpikir dan kemudian dihubungkan dengan kaidah penggunaan bahasa dan aturan dalam pemrograman.

Aplikasi konversi sangat diperlukan dalam kehidupan sehari-hari, antara lain konversi nilai mata uang sesuai kurs yang sedang berlaku, konversi suhu, konversi berat, konversi kecepatan, konversi waktu, dan masih banyak lagi.

Untuk memudahkan pemahaman, perhatikan contoh program konversi waktu dengan menggunakan subprogram procedure di bawah ini. (kalian dianjurkan untuk mencobanya agar mengetahui hasilnya)

```
program konversi_detik;
uses crt;
var
    detikin,jam,detik,menit:longint;
    pil:byte;
procedure detiktojam;
begin
    clrscr;
    gotoxy(32,8); write('Detik : ');
    readln(detikin);
    gotoxy(32,11); writeln('Hasil Konversi : ');
    jam:=detikin div 3600;
    gotoxy(32,13); writeln('Jam : ',jam);
    menit:=round(detikin-3600) div (60*jam);
    gotoxy(32,14); writeln('Menit : ',menit);
    detik:=round(detikin-(jam*3600)-menit*60));
    gotoxy(32,15); write('Detik : ',detik);
    readln;
end;
procedure jamtodelik;
begin
    clrscr;
    gotoxy(32,8); write(' Jam : '); readln(jam);
    gotoxy(31,10); write('Menit : '); readln(menit);
    gotoxy(32,12); write('Detik : '); readln(detik);
    writeln;
    gotoxy(32,14); writeln('Hasil Konversi: '');
```

```
detikin:=(jam*3600)+(menit*60)+detik;
gotoxy(32,16); writeln('Detik : ',detikin);
readln;
end;

begin
repeat
    clrscr;
    gotoxy(32,11); writeln(' Menu Utama ');
    gotoxy(32,14); writeln('1. Konversi Dari
Detik');
    gotoxy(32,16); writeln('2. Konversi Ke Detik');
    gotoxy(32,18); writeln('3. Keluar');
    gotoxy(32,21); write('Silakan Pilih Menu : ');
    readln(pil);
    case pil of
        1:detiktojam;
        2:jamtodetik;
    end;
until pil=3;
end.
```

Dalam pengkonversian waktu yang perlu diperhatikan adalah tipe bilangan bulat yang digunakan, karena ranah nilai tipe integer terbatas, maka ada kemungkinan hasil pengubahan jam-menit-detik ke total detik bernilai negatif, sebab nilai  $\text{detikin:}=(\text{jam}\ast 3600)+(\text{menit}\ast 60)+\text{detik}$  berada di luar rentang tipe integer. Tipe longint yang mempunyai ranah yang lebih besar dapat dipakai untuk masalah ini.

Pada program diatas bila kompilasi, maka akan menghasilkan tampilan pertama adalah menu utama, sehingga mengharuskan pengguna untuk memilih. Jika nomor 1 yang dipilih, maka pengguna harus memasukkan bilangan (detik) untuk kemudian program akan menampilkan bahwa detik tersebut adalah ... jam ... menit ... detik. Jika nomor 2 yang dipilih, maka pengguna harus memasukkan bilangan jam, menit, dan detik, sebagai contoh 6 jam 5 menit 3 detik. Maka program kemudian akan menampilkan 21903 detik. Aplikasi ini tentunya sangat berguna bagi para pelatih olahraga dan para atlet yang membutuhkan kecepatan. Apakah permasalahan lain juga bisa dijadikan program yang menggunakan *procedure*?

### 1.2.2.2. Menanya

Berdasarkan kegiatan mengamati, ada beberapa hal yang bisa diperhatikan dan dibahas lebih lanjut. Beberapa pertanyaan yang berkaitan dengan pemanfaatan prosedur adalah:

1. Bagaimana cara membuat procedure untuk program konversi yang lain?
2. Bagaimana cara membuat procedure pada aplikasi bisnis?
3. Bagaimana cara membuat procedure pada palindrom?

Untuk menjawab pertanyaan-pertanyaan tersebut, siswa diharapkan bisa melakukan kegiatan mencoba berikut dengan bimbingan guru.

### 1.2.2.3 Mencoba

#### Percobaan 1

Untuk mengetahui jawaban dari pertanyaan pertama yaitu cara membuat procedure untuk program konversi yang lain ,salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```
program Konversi_Suhu;
uses crt;
var f,c,r : real;
    a,ul : char;
procedure farein_celcius;
begin
    Writeln('Program Konversi Fareinheit Ke Celcius');
    Writeln('=====');
    Writeln;
    Write('Masukan Suhu dalam Farenheit: ');readln(f);
    c:=5/9*(f-32);
    Writeln;
    Writeln('Jadi Suhu Dalam Celcius Adalah: ',c:4:2);
end;
procedure farein_reamur;
begin
    Writeln('Program Konversi Fareinheit Ke Reamur');
    Writeln('=====');
    Writeln;
    Write('Masukan Suhu dalam Farenheit: ');readln(f);
    r:=4/9*(f-32);
    Writeln;
```

```
        Writeln('Jadi Suhu Dalam Reamur Adalah: ',r:4:2);
end;

procedure celcius_farein;
begin
    Writeln('Program Konversi Celcius Ke Fareinheit');
    Writeln('=====');
    Writeln;
    Write('Masukan Suhu dalam Celcius: ');readln(c);
    f:=(9/5)*c+32;
    Writeln;
    Writeln('Jadi Suhu Dalam Fareinheit Adalah: ',f:4:2);
end;

procedure celcius_reamur;
begin
    Writeln('Program Konversi Celcius ke Reamur');
    Writeln('=====');
    Writeln;
    Write('Masukan Suhu dalam Celcius: ');readln(c);
    r:=(4/5)*c;
    Writeln;
    Writeln('Jadi Suhu Dalam Reamur Adalah: ',r:4:2);
end;

procedure reamur_celcius;
begin
    writeln('Program Konversi Reamur ke Celcius');
    Writeln('=====');
    Writeln;
    Write('Masukan Suhu dalam Reamur: ');readln(r);
    c:=(5/4)*r;
    Writeln;
    Writeln('Jadi Suhu Dalam Celcius Adalah: ',c:4:2);
end;

procedure reamur_farein;
begin
    writeln('Program Konversi Reamur ke Fareinheit');
    Writeln('=====');
    Writeln;
    Write('Masukan Suhu dalam Reamur: ');readln(r);
    f:= (9/4)*r+32;
    Writeln;
```

```
Writeln('Jadi Suhu Dalam Fareinheit Adalah: ',f:4:2);
end;

begin
repeat
    clrscr;
    writeln ('Program konversi suhu');
    writeln;
    writeln ('1. fareinheit - celcius');
    writeln ('2. fareinheit - reamur');
    writeln ('3. celcius - reamur');
    writeln ('4. celcius - fareinheit');
    writeln ('5. reamur - celcius');
    writeln ('6. reamur - farenheit');
    writeln;
    write ('pilih nomor konversi : '); read (a);
    writeln;
    case a of
        '1' : farein_celcius;
        '2' : farein_reamur;
        '3' : celcius_reamur;
        '4' : celcius_farein;
        '5' : reamur_celcius;
        '6' : reamur_farein;
    else
        writeln ('Nomor yang anda masukkan salah');
    end;
    Writeln;
    Writeln;
    Write('Mau Coba Lagi [Y/T]: ');Readln(ul);
    Until Upcase(ul) = 'Y';
End.
```

**Hasil Percobaan Program 1:**

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

**Percobaan 2**

Buatlah program procedure yang digunakan untuk menghitung bunga simpanan dari suatu tabungan, dengan ketentuan : bungan 1 % diberikan jika tabungan kutang dari Rp 100.000,00. Bunga 10 % diberikan jika besar tabungan berada diantara Rp 100.000,00 – Rp 500.000,00. Bunga 15 % diberikan jika besar tabungan berada diatas Rp 500.000,00.

*Salinlah program ke dalam kotak ini.*

**Percobaan 3**

Buatlah program dengan procedure untuk mengoreksi apakah suatu kata yang dimasukkan merupakan palindrom atau bukan, sekaligus dengan ketentuan Uppercase dan Lowercase sehingga palindrom yang ditunjukkan benar-benar kata yang bisa dibalik sama persis dengan kata masukan.

*Salinlah program ke dalam kotak ini.*

**1.2.2.4 Mengasosiasi**

*Pada Percobaan 1*

Program konversi suhu ini dibuat menggunakan prosedur. Variabelnya menggunakan variabel global karena semua variabel di prosedur maupun di program utama sama.

Program ini menggunakan *repeat until* di program utama supaya program ini dapat diulang berulangkali sampai pengguna mengetikkan huruf T.

Dalam percabangannya program menggunakan *case of* karena lebih singkat daripada *if then else*. Kalau menggunakan *if then else* percabangannya bentuknya seperti ini:

```
if a='1' then
    farein_celcius
else if a='2' then
    farein_reamur
else if a='3' then
```

```

        celcius_reamur
    else if a='4' then
        celcius_farein
    else if a='5' then
        reamur_celcius
    else if a='6' then
        reamur_farein;

```

Tentu saja program ini terlalu panjang, maka disini menggunakan *case of* untuk percabangan atau pemilihannya. Dalam pemilihannya angka berada diantara petik satu karena variabelnya dalam bentuk `char`, bila variabelnya dalam bentuk `integer` maka angka-angkanya tidak perlu diberi tanda petik, saya menggunakan `char` karena bitnya lebih rendah.

### **Logikanya :**

Pertama program akan menampilkan :

```

Program konversi suhu
1. fareinheit - celcius
2. fareinheit - reamur
3. celcius - reamur
4. celcius - fareinheit
5. reamur - celcius
6. reamur - farenheit
Pilih nomor konversi :

```

Nomor konversi kita isikan dengan angka 1-6 untuk memilih program konversi  
Bila kita memasukkan angka 5 maka yang keluar adalah program konversi reamur ke celcius.

Berikut tampilannya :

```

Program Konversi Reamur ke Celcius
=====
Masukan Suhu dalam Reamur : 40
Jadi Suhu Dalam Celcius adalah: 50.00
Mau Coba Lagi [Y/T] :

```

Bila kita ketikkan `y`, maka program akan mulai lagi seperti yang pertama.

Bila kalian ingin menghentikan program maka ketikkan `t`. setelah itu program akan berhenti.

Mau Coba Lagi [Y/T] :

program ini menggunakan *repeat until* supaya program dapat diulang-ulang selama *ul* tidak sama dengan *t* maka program akan diulangi terus. Ini merupakan salah satu bentuk perulangan (*loop*).

### Pada Percobaan 2

Kita harus mendefinisikan besar simpanan dengan tipe data real, sehingga dapat mencakup banyak angka masukan. Perhatikan pada deklarasi procedure, kita menggunakan *if and then* disana. Ini digunakan mengingat diantara simpanan Rp 100.000,00 – Rp 500.000,00 mendapatkan bunga 10% dari simpanan.

Sedangkan rumus lengkapnya sebagai berikut :

```
procedure proses();
if simpanan < 100000 then
    simpanan := simpanan - 1000
    else if (simpanan >= 100000) and (simpanan < 500000)
then
    simpanan := (10/100 * simpanan) + simpanan
    else
    simpanan := (15/100 * simpanan) + simpanan;
end;
```

### Pada Percobaan 3

Karena uppercase dan lowercase berbeda, sehingga kata ‘Katak’ tidak akan sama dengan ‘katAK’, dan ‘kasur ini rusak’ ini akan sama dengan ‘kasur ini rusak’, maka kita perlu mendefinisikan ketentuan ini dalam procedure pertama :

```
procedure Palindrom(kt : string);
var
    status : boolean;
    pj,i,j : integer;
```

```

begin
    pj := length(kt);
    for i:=1 to pj do
        kt[i] := upcase(kt[i]);
    i :=1;
    j :=pj;
    status := false;
    while (j>=i) do
    begin
        if(kt[i]=kt[j]) then status := true
        else
        begin
            status := false;
        end;
        i:= i+1;
        j := j-1;
    end;

```

Sedangkan pada program utama akan memanggil procedure tadi jika ternyata TRUE, yaitu

```

if Palindrom(kata) = true then writeln(kata,' adalah Palindrom')
else writeln(kata , 'Bukan Palindrom');

```

Sebagai programer, kita disarankan untuk meminimalisir penggunaan perintah `break`; sehingga pada kasus ini, penggunaan logika proses berpikir sangat diperlukan agar program berhenti sesuai keinginan kita, yaitu dengan `end;` dengan membagi beberapa poin masalah.

Ini adalah pengecekan menggunakan cara pertama, untuk cara kedua silakan coba analogikan sendiri, misalkan dengan mengecek satu demi satu huruf dalam kata tersebut, dari depan dan dari belakang secara bersamaan dan menghitung apakah jumlah karakternya ganjil atau genap. Sehingga ketika salah satu proses tidak sama, maka program akan berhenti untuk kemudian menghasilkan ‘Bukan Palindrom’.

### 1.2.3. Rangkuman

Dari percobaan-percobaan yang telah dilakukan, bisa ambil kesimpulan bahwa:

1. Program yang sebenarnya bisa dilakukan dengan cara biasa, namun dalam penggunaan kata terkadang procedure, maka program akan diketikkan dengan lebih ringkas, karena dapat dipilih mana proses yang utama, mana proses yang membentuk suatu perhitungan, sehingga pada akhirnya procedure perhitungan akan dipanggil dalam procedure hasil dan procedure hasil akan dipanggil dalam program keluaran.
2. Dalam membagi kasus untuk diletakkan dalam procedure, kita diharuskan mengenali logikanya, sehingga programer tidak akan kesulitan dalam proses pemanggilan data.
3. Parameter dan variabel dalam procedure berperan sangat penting dalam pendeklarasian, karena ketika salah meletakkan variabel global atau variabel lokal, maka program akan mismatch atau error.
4. Penggunaan tipe data yang tepat akan memudahkan dalam masukan dan keluaran seperti yang diinginkan pengguna, tentu kita harus tahu jenis data apa yang tepat dan batasan keluaran yang dihasilkan data tersebut.

### 1.2.4. Tugas

Setelah mengikuti kegiatan belajar di atas, berikutnya siswa bisa memperdalam pengetahuannya dan berlatih membuat program sendiri untuk mengasah kemampuan pembuatan program yang memanfaatkan prosedur untuk memecahkan masalah.

1. Tulislah prosedur untuk menghitung jumlah N buah bilangan genap pertama (bilangan genap dimulai dari 0). Prosedur menerima (parameter) masukan N dan memberikan (parameter) keluaran jumlah N buah bilangan genap pertama.
2. Tulislah prosedur yang menghasilkan nilai rata-rata sekumpulan data bilangan bulat yang dibaca secara berulang-ulang dari papan ketik (akhir pembacaan adalah 9999).
3. Ulangi soal nomor 2 tetapi prosedur menghasilkan nilai terkecil.
4. Misalkan Anda menyimpan uang di bank konvensional sejumlah A rupiah pada awal tahun. Jika Anda mendapat bunga tahunan sebesar i persen, maka jumlah uang Anda setelah N tahun adalah

$$F = A \{ (1+i/100) + (1+i/100)^2 + (1+i/100)^3 + \dots + (1+i/100)^n \}$$

Buatlah prosedur yang menerima masukan A,i,n dan memberikan keluaran F.

5. Tulislah prosedur dalam bahasa Pascal yang menerima masukan berupa nilai integer positif dan menampilkan nilai tersebut dalam kata-kata.

*Contoh :*

Masukan :15

Keluaran : lima belas

Masukan : 2347

Keluaran : dua ribu tiga ratus empat puluh tujuh

6. Tuliskan prosedur yang menerima nama hari sekarang dan menentukan nama hari besok. Misalnya, jika hari sekarang “rabu”, maka hari besok adalah “kamis”.
7. Ulangi nomor 6, tetapi menentukan nama hari sebelumnya.
8. Tulislah prosedur yang menerima sebuah tanggal dalam bentuk dd-mm-yyyy (contoh : 12-11-2014) dan memberikan keluaran tanggal sebelumnya. Catatan : parameter tanggal berjenis masukan/keluaran.
9. Tuliskan prosedur yang menerima sebuah tanggal (dd-mm-yyyy) lalu menghitung berapa hari jarak tanggal tersebut dari tanggal 1-1-1900
10. Tulislah prosedur yang menerima jam sekarang (hh:mm:ss), tanggal (dd-mm-yyyy), dan nama hari, kemudian jam terus berputas detik demi detik sehingga ketika mencapai pukul 00:00:00, tanggal berikutnya juga berubah, begitu pula nama hari berikutnya. Perhatikan kasus tahun kabisat.
11. Dibaca nama karyawan dan gaji pokok bulanannya. Gaji bersih yang diterima pegawai adalah: gaji bersih = gaji pokok + tunjangan – pajak
12. Tunjangan karyawan dihitung 20% dari gaji pokok, sedangkan pajak adalah 15% dari gaji pokok ditambah tunjangan. Nama karyawan dan gaji bersihnya dicetak ke piranti keluaran. Buatlah program menggunakan procedure dari masalah tersebut.

### 1.3. Kegiatan Belajar 3. Deklarasi dan Definisi Fungsi

Alokasi Waktu : 2 x 45 menit

#### 1.3.1. Tujuan Pembelajaran

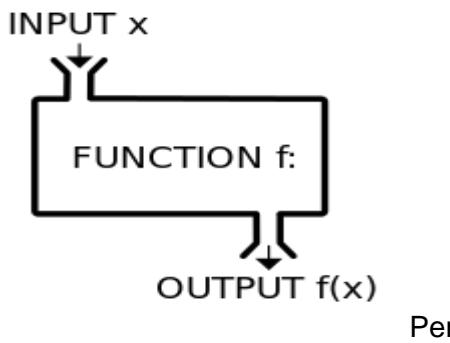
Setelah mengikuti kegiatan belajar 3, siswa diharapkan dapat :

1. Menjelaskan definisi function.
2. Mendeklarasikan function.
3. Menyelesaikan masalah sederhana menggunakan function.

#### 1.3.2. Aktivitas belajar siswa

##### 1.3.2.1 Mengamati/ observasi

Dalam suatu kalkulator terdapat proses perhitungan dengan operasi tertentu yang diinginkan pengguna setelah memberikan masukan berupa angka, kemudian memilih jenis operasi tertentu untuk diolah kepada angka tertentu, selanjutnya akan memberikan keluaran sesuai dengan perhitungan jika dilakukan secara manual.



**Gambar 1.** Ilustrasi fungsi pada kalkulator (gambar fungsi : en.wikipedia.org, kalkulator : v-beauty.co.id)

Secara umum, aturan penulisan deklarasi fungsi (function) sebagai berikut :

```

function Nama_Fungsi(param_formal:tipe_param,...):tipe_hasil;
var
{bagian deklarasi fungsi, sifatnya lokal}
begin Statement-1;
    Statement-2;
    .....
    Statement-n;
    Nama_Fungsi:=hasil;
end;
  
```

Pascal mempunyai beberapa fungsi standar, antara lain Abs, Sqrt, Exp, Ln dan lain-lain. Fungsi tersebut dikatakan standar karena memang sudah dibuat dan tersimpan di dalam *Compiler FreePascal*. Pada saat diperlukan, programmer tinggal memanggil saja.

Selain menyediakan fungsi-fungsi standar, FreePascal memberi fasilitas kepada programmer untuk menciptakan fungsi sendiri, fungsi-fungsi ini disebut **FUNGSI BUATAN (User Defined Function)**. Pada dasarnya, cara menciptakan suatu fungsi sama dengan cara membuat prosedur (yang telah dibahas pada pertemuan sebelumnya). Struktur keduanya sama, perbedaan antara prosedur dengan fungsi buatan hanya terletak pada sistem pertukaran data serta tata cara pemberian namanya saja. Seperti halnya fungsi standar, fungsi buatan dapat menerima beberapa data, tetapi hanya dapat menghasilkan satu keluaran (*output*) saja.

Fungsi buatan diletakkan sesudah bagian deklarasi (konstanta, tipe variabel) tetapi sebelum blok program utama. Programmer dapat mendeklarasikan fungsi pada daerah yang sama dengan deklarasi prosedur.

Fungsi buatan yang sudah diciptakan dapat dipanggil pada berada di blok program utama, dengan cara menuliskan nama fungsi tersebut, diikuti dengan argumen-argumen yang akan dimasukkan. Pada saat fungsi itu dipanggil, perjalanan program akan meloncat ke awal fungsi yang dimaksud. Data-data yang ada di dalam parameter akan diolah sehingga diperoleh sebuah hasil. Hasil tersebut akan dibawa kembali ke tempat dimana fungsi tersebut dipanggil.

Contoh program dengan fungsi buatan :

```
program fungsii;
function tambah(X, Y : integer) : integer;
begin
    Tambah := X+Y;
end;
{program utama}
begin
    writeln('2+3 = ', tambah(2,3));
end.
```

Berikut ini beberapa hal yang harus diperhatikan pada saat menciptakan fungsi buatan sendiri.

- Fungsi harus diberi judul. Judul suatu fungsi diawali dengan kata `Function` dan diikuti nama fungsi yang akan dibuat, serta jenis data yang akan dihasilkan oleh fungsi tersebut. Nama suatu fungsi harus diawali dengan huruf/abjad, tidak mengandung spasi, dan tidak lebih dari 63 karakter. Selain itu, nama suatu fungsi harus sama dengan nama variabel yang akan digunakan untuk menyimpan proses. Setelah menuliskan nama fungsi, programmer harus menyebutkan semua parameter yang akan digunakan oleh fungsi tersebut beserta tipenya (di dalam kurung), fungsi buatan harus memiliki parameter. Parameter pada fungsi buatan dapat disamakan dengan argumen pada fungsi standar. Parameter akan diisi dengan data-data mentah yang akan diolah oleh fungsi yang bersangkutan. Setelah semua parameter disebutkan, terdapat dua titik yang dilanjutkan dengan jenis data yang akan dihasilkan oleh fungsi tersebut. Contoh :

```
FUNCTION VOLUME(p,l,t:integer):integer;
```

- Suatu fungsi dapat memiliki beberapa argumen, tetapi hanya dapat mengeluarkan satu hasil saja. Data yang dihasilkan oleh sebuah fungsi harus disimpan di dalam variabel yang namanya sama dengan argumen tersebut.

## Fungsi Berganda

Seperti halnya prosedur, suatu fungsi dapat dipanggil ketika sedang berada dalam fungsi yang lain, tentu saja fungsi yang dipanggil tersebut harus sudah ada, dibuat dahulu sebelumnya.

### 1.3.2.2 Menanya

Berdasarkan kegiatan mengamati, ada beberapa hal yang bisa diperhatikan dan dibahas lebih lanjut. Beberapa pertanyaan yang berkaitan dengan fungsi adalah:

- 1 Bagaimana cara membuat program dengan function tanpa menggunakan parameter?
- 2 Bagaimana cara membuat program dengan menggunakan satu parameter?
- 3 Bagaimana cara membuat program dengan menggunakan lebih dari satu parameter?
- 4 Apa perbedaan antara function dengan procedure?
- 5 Bagaimana cara menggunakan function dalam penyelesaian masalah?

Untuk menjawab pertanyaan-pertanyaan tersebut, siswa diharapkan bisa melakukan kegiatan mencoba berikut dengan bimbingan guru.

### 1.3.2.3 Mencoba

#### Percobaan 1

Untuk mengetahui jawaban dari pertanyaan pertama yaitu cara membuat program dengan function tanpa menggunakan parameter, pada awal bab ini kalian sudah mempelajari bagaimana membuat *procedure* dari program ‘Mencari Luas dan Keliling Persegi Panjang’, coba sekarang buatlah dengan menggunakan *function* dari program dasar berikut :

```
Program PersegiPanjang;                                {Judul Program Utama}
uses crt;sa
var
  p,l:byte;                                         {deklarasi variabel global}
  ls,kel:integer;                                    {p :panjang, l:lebar}
begin
  clrscr;
  writeln('luas dan keliling persegi panjang');
  writeln;
  write('Masukan panjang : ');readln(p);
  write('Masukan lebar   : ');readln(l);
  writeln;
  ls:=p*l;                                         {luas=panjang x lebar}
  Kel:=(2*p)+(2*l);                                {keliling=(2 x panjang) + (2 x lebar)}
  writeln('Luas      : ',ls);
```

```
writeln('Keliling : ',kel);
readln;
end.
```

### Hasil Percobaan Program PersegiPanjang:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Salin pesan asli kompilasi disini:

.....  
.....  
.....

Letak kesalahan (baris, kolom)	Terjadi karena

Perhatikan fungsi berikut yang digunakan untuk mengkonversi nilai mata uang dalam rupiah ke dolar. Silakan isi titik-titik yang masih kosong kemudian cek pada saat kompilasi, apakah sudah benar? Bagaimana jika ingin mengkonversi rupiah ke dalam beberapa mata uang (tanpa menggunakan tipe data array)? Apakah bisa? Jika tidak bisa, hal apa yang menyebabkan program tersebut error?

### Percobaan 2

```
program KursUang;
var
  rupiah,kursdolar : real;
//isi dengan tipe data yg sesuai
  dolar: real;
  function nilaidolar(rupiah,kursdolar: integer): real;
//isi dengan nama fungsi yg sesuai
```

```

begin
    nilaidolar := rupiah/kursdolar;
    //isi dengan pendefinisian perhitungan yang sesuai
end;

begin
write('masukkan nilai rupiah = ');readln('rupiah') ;
    //isi dengan parameter yang sesuai
write('kurs 1 dolar = '); readln(kursdolar);
    dolar:=nilaidolar(rupiah,kursdolar);
    //isi dengan nama fungsi yang sesuai
write(dolar,'dolar');
end.

```

### Hasil Percobaan Program KursUang:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

### Percobaan 3

Perhatikan program berikut ini. Berikut adalah program yang terdapat dua buah pendeklarasian function dan saling berhubungan, coba ubahlah menjadi program yang dideklarasikan dengan menyisipkan fungsi penjumlahan di dalam fungsi ratarata!

```

program FungsiBerganda;
var
    a, b, c : Integer;
    function Jumlah (x, y, z : Integer) : Integer;

```

```

begin
    Jumlah := x+y+z;
end;

function ratarata (x,y,z : Integer) : Integer;
var
    sum : Integer;
begin
    sum := jumlah (x,y,z);
    ratarata := sum/3;
end;

begin
write ('Tuliskan tiga bilangan a,b,c : ');
readln (a,b,c);
writeln ('Rata-ratanya = ', ratarata (a,b,c));
end.

```

### Hasil percobaan Program FungsiBerganda

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

### Percobaan 4

Coba selidiki apakah program berikut benar atau salah dalam pendeklarasian function, coba jelaskan!

```

Program UjiFungsi;

function TitikTengah(P1, P2 : Titik):Titik;
begin
    TitikTengah.x := (P1.x + P2.x)/2;
    TitikTengah.y := (P1.y + P2.y)/2;
end;

```

```
begin  
end.
```

**Hasil percobaan program UjiFungsi :**

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Salin pesan asli kompilasi disini:

.....  
.....  
.....

Letak kesalahan (baris, kolom)	Terjadi karena

**1.3.2.4 Mengasosiasi/ menalar***Pada percobaan 1*

Pada saat membuat suatu program yang kemudian diubah ke dalam bentuk program yang menggunakan procedure dan function, terdapat beberapa perbedaan dan kesamaan dalam hal pendeklarasiannya. Sama seperti *procedure*, untuk mengakses setiap function kita harus memanggil namanya. *Function* juga dapat memiliki parameter, tetapi parameter yang terdapat dalam *function* selalu merupakan parameter masukan (*input parameter*). Tipe hasil di dalam suatu fungsi merupakan tipe nilai balik (*return*) yang diberikan oleh fungsi. Bila kita perhatikan *listing program* diatas, terdapat sebuah function dengan nama *Luas* yang memiliki parameter *p* dan *l*, sedangkan tipe nilai balik (*return*) yang dihasilkan oleh fungsi tersebut adalah *integer*. Pada percobaan 1 diatas *Luas* digunakan sebagai nama fungsi, dan didalam fungsi tersebut, *Luas* digunakan sebagai variabel untuk menampung hasil perhitungan pada fungsi tersebut.

Dari percobaan 1 diperoleh kesimpulan bahwa suatu program tentu tidak harus dideklarasikan dengan menggunakan *procedure* atau *function*, tergantung pembuat program apakah ingin lebih menghemat waktu dalam penulisan program dan mempermudah pengecekan apabila terdapat kesalahan atau ingin menjelaskan program yang berjalan secara sederhana dengan tidak menggunakan *function* dan atau *procedure*.

### Pada Percobaan 2

Dari percobaan di atas, apabila titik-titik pada persoalan tersebut diisi dengan :

```
isi1 'integer'  
isi2 'nilaidolar'  
isi3 'rupiah/kursdolar,'  
isi4 '(rupiah)'  
isi5 'nilaidolar'
```

dan kemudian dikompilasi (Alt+F9) maka akan menampilkan bahwa *source code* yang dituliskan bisa dibaca komputer. Dari percobaan pengkonversian rupiah ke dollar menggunakan tiga variabel yaitu rupiah, kurs dolar, dan dolar dan mempunyai satu keluaran, inilah yang merupakan ciri *function* dan *procedure*, untuk mencari pengkonversian rupiah ke dalam mata uang asing lain, diperlukan pendefinisian baru terhadap variabelnya, misalnya kita ingin mengkonversi rupiah ke dalam dolar amerika, dolar singapura dan ringgit malaysia, sehingga variabelnya antara lain : rupiah, kursdolaramerika, kursdolarsingapura, ringgitmalaysia, yang berupa integer, kemudian dollaras, dollarsg, ringgitmy yang berupa real.

### Pada Percobaan 3

Perhatikan bahwa fungsi jumlah terletak di dalam fungsi rata-rata. Oleh karena itu, fungsi jumlah tersebut memiliki ruang lingkup lokal terhadap fungsi rata-rata. Fungsi jumlah hanya dapat dipanggil dari dalam blok fungsi rata-rata. Kita dapat memanggil fungsi jumlah dari bagian utama program.

Fungsi jumlah hanya akan dijalankan apabila fungsi rata-rata dijalankan. Kesimpulan yang dapat diperoleh adalah bahwa suatu fungsi hanya dapat dipanggil dari dalam blok tempat fungsi tersebut berada.

Variabel a, b, c adalah variabel global, sedangkan variabel sum adalah variabel lokal yang hanya dikenali oleh blok fungsi rata-rata. Variabel sum dikenali oleh fungsi jumlah sebab fungsi jumlah berada di dalam blok fungsi rata-rata.

```
program fungsi_berganda;
var
    a, b, c : Integer;
    function ratarata (u,v,w : Integer) : real;
    var
        sum : Integer;
    function jumlah (x, y, z : Integer) : Integer;
    begin
        Jumlah := x+y+z;
    end;
    begin
        sum := jumlah (u,v,w);
        ratarata := sum/3;
    end;
begin
    writeln ('Tuliskan tiga bilangan a, b, c : ');
    readln (a, b, c);
    writeln ('Rata-ratanya = ', Ratarata (a, b, c));
end.
```

#### Pada Percobaan 4

Penulisan function pada program tersebut salah. Penulisan fungsi TitikTengah yang mengembalikan nilai bertipe Titik pada fungsi diatas salah, karena fungsi di dalam Pascal hanya dapat mengembalikan nilai yang bertipe dasar (integer, real, char, string dan boolean)

Agar fungsi TitikTengah dapat ditranslasi dalam bahasa Pascal, fungsi tersebut harus dimanipulasi sehingga ia mengembalikan nilai yang bertipe sederhana (misal integer), sementara titik tengah disimpan dalam sebuah *parameter by reference*.

### 1.3.1. Rangkuman

Dari percobaan-percobaan yang telah dilakukan, bisa ambil kesimpulan bahwa:

1. Sebuah fungsi digunakan untuk menghitung sebuah nilai berdasarkan satu atau beberapa nilai masukan. Dalam pascal, fungsi akan membantu mewujudkan pemecahan masalah yang lebih sederhana, sehingga program semakin mudah dibaca dan lebih mudah diubah atau dimodifikasi. Di dalam dunia pemrograman untuk membuat suatu program yang besar, maka program tersebut perlu disusun dari subprogram yang kecil-kecil. Setiap subprogram terkadang dapat berdiri sendiri dari program utama dan tiap subprogram itu disebut sebagai modul program.
2. Function dalam bahasa Pascal tidak dapat mengembalikan nilai yang bertipe terstruktur. Karena itu, fungsi yang mengembalikan tipe terstruktur harus dimanipulasi dengan cara mengubah tipe hasilnya menjadi tipe dasar (misalnya integer atau boolean).
3. Beberapa keuntungan ketika kita memakai fungsi dalam FreePascal :
  - a. Mengurangkan tugas pemrograman rumit menjadi langkah-langkah yang lebih sederhana atau kecil.
  - b. Mengurangi duplikasi kode (kode yang sama ditulis berulang-ulang) dalam program.
  - c. Dapat menggunakan kode yang ditulis dalam berbagai program yang berbeda.
  - d. Memecah program besar menjadi kecil sehingga dapat dikerjakan oleh programmer-programmer atau dipecah menjadi beberapa tahap sehingga mempermudah penggerjaan dalam sebuah projek
  - e. Menyembunyikan informasi dari user sehingga mencegah adanya perbuatan abnormal seperti memodifikasi atau mengubah program yang kita buat
  - f. Meningkatkan kemampuan pelacakan kesalahan, jika terjadi suatu kesalahan kita tinggal mencari fungsi yang bersangkutan saja dan tak perlu mencari kesalahan tersebut di seluruh program.

### 1.3.2. Tugas

Setelah mengikuti kegiatan belajar di atas, berikutnya siswa bisa memperdalam pengetahuannya dan berlatih membuat program sendiri untuk mengasah kemampuan pembuatan program yang memanfaatkan fungsi.

1. Perhatikan beberapa judul fungsi di bawah ini!

Tentukan manakah judul fungsi yang benar!

- a. FUNCTION Maksi;
- b. FUNCTION Isi (A: B: Integer) : Integer;
- c. FUNCTION Isi (A : Boolean) : Char;
- d. FUNCTION Maksimum (A, B : Real :C) : Integer;
- e. FUNCTION ab cd (A : Integer) : Char;

- f. FUNCTION zkali Y (z real, Y real) : Char;
  - g. FUNCTION Pendapatan (Gaji : Real; Kode : String[B]) : String[20];
2. Tuliskan judul fungsi untuk keperluan berikut ini :
- a. Fungsi riil bernama AbagiB dengan dua buah parameter riil a dan b.
  - b. Fungsi integer bernama Terkecil dengan tiga buah parameter integer A, B dan C.
  - c. Fungsi riil bernama Pajak dengan parameter KodeWilayah (Integer), Gaji (riil), dan Status (Character).
3. Buatlah sebuah fungsi untuk menghitung luas permukaan bola dengan rumus :  $L = 4 \pi r^2$
4. Sin adalah fungsi standar yang sudah tersedia untuk menghitung sinus suatu sudut, argumen fungsi ini harus dalam satuan radian. Buatlah program untuk menanyakan sudut dalam derajat lalu menghitung sinus sudut tersebut.
5. Buatlah program untuk menanyakan sisi tegak sebuah segitiga siku-siku, lalu menghitung sisi miringnya dengan menggunakan sebuah fungsi.
6. Buatlah sebuah fungsi riil untuk menghitung rata-rata tiga buah bilangan integer x, y, z.

Lembar Kreativitas Siswa.

Buatlah sebuah fungsi berdasarkan ide dari siswa sendiri.

```
program
var

function
var

begin

end; //akhir dari fungsi

//program utama
begin

end;
```

## 1.4. Kegiatan Belajar 4. Pemanfaatan Prosedur dan Fungsi dalam Aplikasi

**Alokasi Waktu : 2 x 45 Menit**

### 1.4.1. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 4, siswa diharapkan dapat :

1. Membedakan penggunaan prosedur dan fungsi dalam pembuatan program.
2. Menggabungkan prosedur dan fungsi kedalam satu masalah pada permrograman.
3. Menggunakan function dalam aplikasi yang memuat permasalahan bisnis.

### 1.4.2. Aktivitas Belajar Siswa

#### 1.4.2.1. Mengamati/ Observasi

Misalkan terdapat sekumpulan data yang akan digunakan untuk mencari rata-rata nilai persiswa dan mengetahui apakah siswa tersebut lulus atau tidak pada suatu mata pelajaran untuk rekapitulasi nilai raport menggunakan program Pascal. Apa yang akan kita lakukan pertama kali? Perhatikan gambar dibawah ini.

	A	B	C	D
1	No	Nama	Nilai	Keterangan Kelulusan
2	1	Adi	65	LULUS
3	2	Ida	75	LULUS
4	3	Nuri	55	TIDAK LULUS

**Gambar 2.** Data nilai

Pada pertemuan sebelumnya, kita telah mempelajari tentang aturan penggunaan looping, array dan logic decision, kita juga telah mempelajari tentang procedure dan fuction. Lalu bagaimana cara menggunakan procedure dan function secara bersamaan? kasus apa saja yang dapat dijadikan prosedur dan kasus yang menggunakan fungsi, atau kasus yang bisa keduanya dan kasus yang harus memuat keduanya dalam pembuatan program?

#### 1.4.2.2. Menanya

1. Bagaimana cara menggunakan function dalam penerapan pada aplikasi bisnis, mengecek palindrom dalam kata masukan atau masalah lain dalam kehidupan sehari-hari?
2. Bagaimana cara memanggil function di dalam procedure dan juga sebaliknya?

### 1.4.2.3. Mencoba

#### Percobaan 1

Dalam satu kelas XI IPA 3, terdapat empat nilai hasil ulangan harian dalam satu semester, pada mata pelajaran matematika, guru mata pelajaran tersebut sepakat untuk menjadikan nilai 70 sebagai batas kelulusan siswa, jika rata-rata nilai siswa tidak memenuhi batas minimal kelulusan maka dinyatakan tidak lulus pada pelajaran tersebut. Bagaimana membuat program untuk mengetahui siswa mana saja yang dinyatakan lulus dan dinyatakan tidak lulus?

Perhatikan program berikut, isilah titik-titik yang ada, kemudian cobalah pada program FreePascal untuk mengetahui apakah sudah benar dalam kompilasi dan ketika dijalankan? Kemudian jelaskan cara kerja (alur) program tersebut! Permasalahan tersebut bisa dituliskan dalam bentuk fungsi dan prosedur pada Pascal.

```
program Kelulusan;
var
  a,b,c,d : .......;
  // isi dengan tipe data yang sesuai
  function ..... (a,b,c,d : Integer):Real;
  //isi dengan nama_fungsi yang tepat
  begin
    Ratarata := (a+b+c+d)/4;
  end;
  function Lulus(zz : Integer): Boolean
  begin
    If zz>70 then Lulus := True;
    else Lulus := False;
  end;
  procedure Kelulusan (a,b,c,d : Integer);
  var
    ..... : Real;
    //isi nama variabel yang tidak diketahui
    OK : Boolean;
  begin
    NilaiAkhir := Ratarata (a,b,c,d);
    OK := Lulus (Nilai Akhir);
    if OK = True then WriteIn ('Lulus');
    else WriteIn ('Tidak Lulus');
  end;
```

```

begin
    WriteIn ('Tuliskan Nilai1, Nilai2, Nilai3, Nilai4 : ');
    ReadIn (a, b, c, d);
    Kelulusan (a, b, c, d);
end.

```

### Hasil Percobaan Program Kelulusan:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

### Percobaan 2

Dalam sebuah perusahaan, berlaku aturan jika karyawan mendapatkan gaji kurang dari Rp 2.000.000,00 maka akan dikenakan potongan (pajak) 1% untuk biaya koperasi bulanan, dan jika mempunyai gaji lebih dari Rp 2.000.000,00 maka akan dikenakan pajak sebesar 3%. Bagaimana cara membuat program agar bisa mempermudah bendahara menghitung pajak setiap karyawan dalam perusahaan tersebut?

Program yang akan dibuat, dicoba dengan menggunakan function ke dalam procedure.

S

Isilah titik-titik yang ada sesuai petunjuk yang tersedia, kemudian kompilasilah ke program FreePascal, apakah ada error? Jika tidak, Jelaskan alur proses program di bawah ini :

```

program HitungPajak;
var
    gaji : Real;
procedure ..... (a : real);
    //isi nama prosedur yang sesuai

```

```
var
    Persen, pajak : real;
function ..... (zz : real) : real;
//isi nama function yang sesuai
begin
    if zz< ..... then Persentase := 0.01
    //isi data yang sesuai
    else Persentase := ..... ;
    //isi data berupa angka yang sesuai
end;
begin
    Persen := Persentase (a);
    Pajak := a*persen;
    WriteIn ('Besarnya gaji := Rp. ', a);
    WriteIn ('Besarnya Pajak := Rp. ', pajak);
end;
begin
    Write ('Tuliskan gaji anda : ');
    ReadIn (Gaji);
    HitungPajak (gaji);
end.
```

**Hasil Percobaan Program HitungPajak:**

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

**Percobaan 3**

Bagaimana jika mengubah program dari permasalahan percobaan 2 menjadi procedure di dalam function?

Berikut akan disajikan program yang menunjukkan bahwa suatu prosedur dapat diletakkan di dalam fungsi. Isilah titik-titik berikut kemudian kompilasilah (Alt+F9). Bandingkan hasilnya dengan percobaan 2!

```
program HitungPajak2;
var
    Gaji : ....;
    //isilah dengan tipe data yang sesuai
    function Pajak (a : real) : real;
    var
        Persen, Pajak : Real;
        procedure MenghitungPersen;
    begin
        if a < 2000000 then Persen := 0.01
        else Persen := 0.03;
    end;
    begin
        MenghitungPersen (a);
        Pajak := ... * ....;
        //isi dengan rumus yang tepat
    end;
    begin
        Write ('Tuliskan gaji anda : ');
        ReadIn (gaji);
        WriteIn (Besarnya Gaji := Rp. ', a);
        WriteIn ('Besarnya Pajak := Rp. ', Pajak(gaji));
    end.
```

**Hasil Percobaan Program HitungPajak2:**

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

#### Percobaan 4

Salin dan kompilasilah program palindrom berikut ini, kemudian cobalah input kata MaLam, Siang, KAtak, Makan. Apakah kata-kata tersebut merupakan palindrom?

```
program palindrom;
uses crt;
var
  kata : string;
  function Palindrom(kt : string): boolean;
  var {semua variable proses di tempatkan di blok fungsi}
    status : boolean;
    pj,i,j : integer;
begin
  pj := length(kt);
  for i:=1 to pj do
    kt[i] := upcase(kt[i]);
  {perintah UPCASE untuk menjadikan kata yang masuk ke fungsi menjadi huruf capital semua sehingga kasus yang terjadi pada program pertama dapat dihindari }

  i :=1;
  j :=pj;
  status := false;
  while (j>=i) do
  begin
    if(kt[i]=kt[j]) then status := true
```

```
else
begin
    status := false;
end;

i:= i+1;
j := j-1;
end;

Palindrom:= status; {pemberian nilai ke fungsi}
end;

{PROGRAM UTAMA}
begin
    clrscr;
    write('Masukan Sepatah Kata = ');
    readln(kata);
{memanggil fungsi palindrome jika nilai fungsi ketika dipanggil TRUE maka kata adalah palindrom}
    if Palindrom(kata) = true then writeln(kata,' adalah
Palindrom')
    else
        writeln(kata , ' Bukan Palindrom');

    readln;
end.
```

#### Hasil Percobaan Program 4:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

#### 1.4.2.4. Mengasosiasi

##### Pada Percobaan 1

Program menjalankan *function* Ratarata terlebih dahulu yang memproses inputan data a,b,c,d kemudian disimpan. Pada bagian program utama (*procedure*) nilai Ratarata didefinisikan ulang dengan nama NilaiAkhir, untuk baris selanjutnya pada program utama, program membaca definisi Lulus(NilaiAkhir) yang dideklarasikan pada fungsi kedua, dimana pada fungsi kedua kategori Lulus adalah nilai Ratarata yang dimasukkan kedalam parameter zz, kemudian dideklarasikan jika lebih dari 70 akan disebut ulang dengan ‘Lulus’ untuk dikombinasikan dengan fungsi pertama pada program utama, Jika hasil dari fungsi kedua tadi ‘Lulus’, maka program akan menampilkan bahwa rata-rata tersebut termasuk ‘Lulus’, jika tidak terpenuhi maka program akan menampilkan ‘Tidak Lulus’.

Dari program penjelasan program tersebut, kita tahu bahwa dengan membagi kasus menjadi beberapa bagian dengan menggunakan function, kemudian dikombinasikan ulang dan hasil dari beberapa function tersebut dipanggil pada perintah *procedure*. Dengan demikian kita ternyata bisa mendeklarasikan *function* ke dalam *function*.

##### Pada Percobaan 2

Berbeda dengan kasus pada percobaan 1, fungsi persentase terletak di dalam prosedur HitungPajak, sehingga fungsi persentase hanya dikenali oleh blok prosedur HitungPajak saja, fungsi persentase tidak dapat dipanggil oleh bagian utama program. Gaji adalah variabel global, sedang persen dan pajak adalah variabel lokal yang hanya dikenali oleh blok prosedur hitungpajak. Fungsi persentase dapat mengenali variabel persen dan pajak karena fungsi persentase berada di dalam blok prosedur hitungpajak.

##### Pada Percobaan 3

Prosedur menghitungpersen terletak didalam fungsi pajak, sehingga prosedur menghitungpersen hanya dikenali oleh blok fungsi pajak saja. Bagian utama program tidak dapat memanggil prosedur menghitung persen. Gaji adalah variabel global, sedang persen dan pajak adalah vaeriabel lokal yang hanya dikenali oleh blok fungsi pajak saja. Kedua variabel lokal tersebut dapat dikenali oleh prosedur menghitungpersen sebab prosedur menghitungpersen berada di dalam blok fungsi pajak.

Dengan hasil pada percobaan 2 dan 3, dapat kita ketahui bahwa suatu permasalahan dapat diselesaikan dengan menggunakan 2 subprogram sekaligus, dan dari program pajak1 dan pajak2 dapat kita simpulkan bahwa ternyata kita bisa menyisipkan *function* pada

*procedure* dan menyisipkan *procedure* pada function dengan pendefinisian parameter dan variabel yang tepat maka akan menghasilkan hasil dan kegunaan yang sama.

#### *Pada Percobaan 4*

Program akan merespon kata-kata yang dimasukkan ke dalam masukkan, jika true, maka program akan menganggap bahwa kata tersebut dapat dibalik (huruf besar dan kecil tidak diperhatikan) dan disebut palindrom.

#### **1.4.3. Rangkuman**

1. Tujuan penggunaan function antara lain adalah kode program pada program utama menjadi sederhana (pendek) sehingga mudah untuk di baca dan dimengerti serta variable yang digunakan pada program utama menjadi lebih sedikit dibandingkan program non modular. Karena semua variable untuk keperluan proses ditempatkan pada fungsi.
2. Fungsi dan prosedur dapat digunakan bersama-sama di dalam sebuah program, fungsi dapat diletakkan di dalam suatu prosedur, demikianpula sebaliknya, prosedur dapat diletakkan di dalam suatu fungsi. Selain itu fungsi dan prosedur dapat pula diletakkan terpisah satu sama lain.
3. Pada saat menggunakan fungsi dan prosedur bersama-sama, pembuat program harus selalu mengingat kembali hal sebagai berikut : Ruang lingkup variabel global dan variabel lokal yang dipergunakan. Proses yang dilakukan oleh suatu prosedur dapat menghasilkan lebih dari satu keluaran, sedang proses yang dilakukan oleh fungsi hanya akan menghasilkan satu keluaran saja. Nama suatu prosedur tidak boleh sama dengan variable yang digunakan, tetapi nama suatu fungsi justru harus sama dengan nama variable yang digunakan untuk menyimpan hasil akhir proses.

#### **1.4.4. Tugas**

Setelah mengikuti kegiatan belajar di atas, berikutnya siswa bisa memperdalam pengetahuannya dan berlatih membuat program sendiri untuk mengasah kemampuan pembuatan program yang memanfaatkan fungsi.

Buatlah program untuk menghitung nilai akhir siswa dengan menampilkan “nilai huruf” dari “nilai akhir”. Pertama, buatlah fungsi “NilaiAkhir” yang digunakan untuk memproses perhitungan agar mendapatkan nilai akhir.

Ketentuan untuk mendapatkan nilai akhir seperti berikut:

$$\text{Nilai Murni Absensi} = \text{Nilai Absen} * 30\%$$

$$\text{Nilai Murni Tugas} = \text{Nilai Tugas} * 30\%$$

$$\text{Nilai Murni Ujian} = \text{Nilai Ujian} * 40\%$$

$$\text{Nilai Akhir} = \text{Nilai-Murni Nilai Murni diatas}$$

Kemudian buatlah fungsi “NilaiGrade” untuk memproses perhitungan agar mendapatkan nilai grade. Ketentuan untuk mendapatkan Nilai Grade seperti berikut:

$$\text{Nilai Murni} \geq 80, \text{Nilai Grade} = A$$

$$\text{Nilai Murni} \geq 70, \text{Nilai Grade} = B$$

$$\text{Nilai Murni} < 70, \text{Nilai Grade} = C$$

#### 1.4.5. Uji Kompetensi

1. Deklarasi prosedur manakah yang dibenarkan?
  - a. procedure hapus;
  - b. procedure hapus(s:string);
  - c. procedure hapus(var s:string);
  - d. procedure hapus(s:string):boolean;
  - e. procedure hapus(var data);
2. Deklarasi function manakah yang tidak diizinkan?
  - a. Function density(x:real):real;
  - b. Function density(b:byte):byte;
  - c. Function density(var s:string):real;
  - d. Function density(var data):byte;
  - e. Function density;
3. Tipe variabel ekspresi manakah yang tidak dapat ditampilkan dengan procedure Writeln?
  - a. Type T=integer;
  - b. Type T=string;
  - c. Type C=char;
  - d. Type T=(Small, Medium, Large)
  - e. Semua valid

4. Diberikan program

```
var s:string;
begin
  s:='SMK GO GET GOLD!';
  delete(s,1,length(s)-12);
  writeln(s);
end.
```

Apa keluaran program di atas ?

- a. GO GET GOLD!
- b. GO GET GOLD!
- c. GET GOLD!
- d. SMK GO GET
- e. SMK GO GE

5. Diberikan program

```
var i,k: integer;
begin
  i:=5; k:=0;
  k:=trunc(sqrt(i))+1;
  writeln(k);
end.
```

Apa keluaran program di atas?

- a. 3
- b. 2.24
- c. 2
- d. 0
- e. program tidak dapat dijalankan

6. Mengacu pada program berikut :

```
var
A,B:string;
C:string[10];
begin
  A:='SMK';
  B:='BISA';
  C:=A+B;
  if (Pos(B)>0) then
  Begin
    Writeln('A');
  end else
    Writeln('B');
end.
```

Apa yang terjadi jika program di atas di jalankan...

- a. Huruf 'A' tercetak
- b. Huruf 'B' tercetak
- c. Tidak dapat dipastikan
- d. Terjadi error
- e. Tidak bisa di compile

7. Perhatikan potongan program berikut :

```
begin  
writeln(round(frac(3.7)));  
end.
```

Apa keluaran program di atas ?

- a. 0
- b. 1
- c. 2
- d. 3
- e. 4

8. Perhatikan program berikut

```
var St: String;  
procedure Sulap(var S: String);  
begin  
    if S = 'Kecil' then S := 'kecil' else  
        if S = 'Besar' then S := 'BESAR';  
end;
```

Di antara potongan program berikut, manakah yang salah?

- a. St := Chr(60);  
Sulap(St);
- b. St := 'KECIL';  
Sulap(St);
- c. St := Chr(45) + Chr(65);  
Sulap(St);
- d. Sulap('Besar');
- e. Semua ekspresi di atas benar

9. Apa guna parameter dalam subprogram?

- 10. Kapan kita menggunakan fungsi dan kapan pula menggunakan prosedur?
- 11. Buatlah program untuk menghitung perkalian dua bilangan kompleks! Gunakan prosedur atau fungsi.
- 12. Jelaskan pengertian procedure dan struktur pendeklarasiannya.

## 1.5. Kegiatan Belajar 5. Fungsi Rekursif dan Aplikasinya

**Alokasi Waktu : 3 x 45 menit**

### 1.5.1. Tujuan Pembelajaran

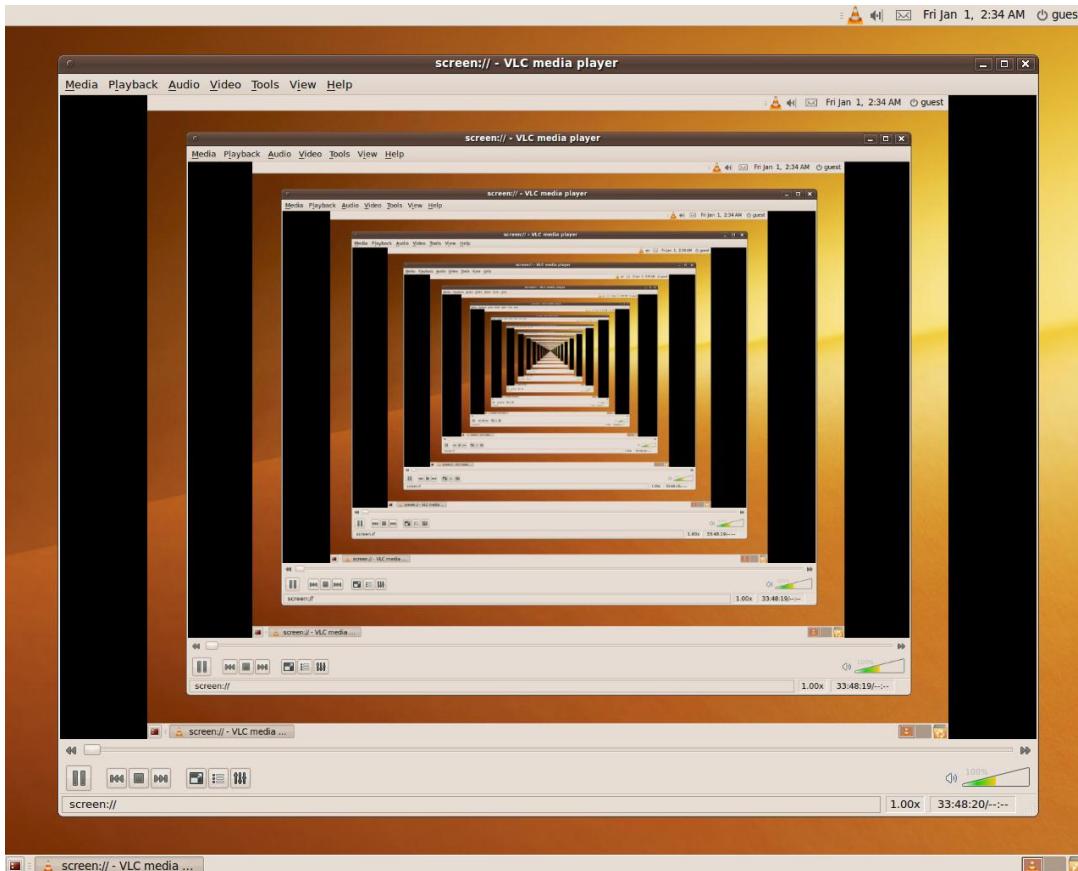
Tujuan pembelajaran pada Kegiatan Belajar 1 tentang Fungsi Rekursif adalah:

1. Siswa dapat memahami Konsep Fungsi Rekursif
2. Siswa dapat mengaplikasikan Fungsi Rekursif dalam menyelesaikan masalah
3. Siswa dapat membuat program sederhana dengan Fungsi Rekursif

### 1.5.2. Aktivitas belajar siswa

#### 1.5.2.1. Mengamati/ observasi

##### Definisi Fungsi Rekursif



**Gambar 3.** Bentuk Rekursif (<http://andiagusta.blogspot.com>)

Perhatikan Gambar di atas! Dalam gambar ini ada gambar ini lagi yang di dalamnya ada gambar ini lagi dan seterusnya sampai kedalaman tak terbatas. Mari kita lihat contoh lain rekursif yang jauh lebih sederhana dan lebih mengenyangkan. Masalah yang akan dipecahkan adalah memotong roti tawar tipis-tipis sampai habis. Jika masalah ini akan dipecahkan secara rekursif, maka solusinya adalah:

1. Jika roti sudah habis atau potongannya sudah paling tipis, pemotongan roti selesai

2. Jika roti masih bisa dipotong, potong tipis dari tepi roti tersebut, lalu lakukan prosedur 1 dan 2 untuk sisa potongannya.

Itulah contoh konsep rekursif dalam kehidupan sehari - hari. Rekursif adalah suatu proses yang bisa memanggil dirinya sendiri berupa statement perulangan. Dalam pemrograman, rekursi merupakan teknik pemrograman yang penting, dan beberapa bahasa pemrograman modern mendukung keberadaan proses rekursi ini termasuk pascal.

Adapun Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri selama kondisi pemanggilan dipenuhi. Dari definisi tersebut maka fungsi rekursif disusun oleh dua bagian:

1. Basis, Bagian yang berisi kasus yang terdefinisi secara eksplisit atau langsung, Bagian ini juga sekaligus menghentikan rekursif dan memberikan sebuah nilai yang terdefinisi pada fungsi rekursif
2. Rekurens, Bagian ini menyebabkan pemanggilan diri fungsi karena kondisi khusus tidak dipenuhi

Fungsi 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 fungsi rekursif menjadi pilihan kita bila memang memungkinkan untuk dipergunakan.

### **Skema Umum Fungsi Rekursif**

Secara umum skema fungsi rekursif adalah sebagai berikut: :

```
Function NamaFungsi(Input_x : Tipe_x) : Tipe_Fungsi;
//Algoritma
Begin
  If  T(Input_x)  then N(Input_x)           {Basis}
  Else NamaFungsi(g(Input_x));             {Rekurens}
End;
```

Keterangan:

NamaFungsi Adalah nama fungsi yang digunakan

Input\_x Adalah parameter fungsi yang digunakan

Tipe\_x Adalah tipe data dari parameter x

Tipe\_Fungsi Adalah tipe data dari fungsinya

T Adalah fungsi untuk kasus basis

N Adalah fungsi jika kasus basis tercapai

g Adalah fungsi yang mengubah nilai parameter x

Contoh 1. Faktorial

Fungsi faktorial adalah contoh fungsi yang rekursif, fungsi rekursif ini mengandung dirinya sendiri dalam definisinya:

$$0! = 1$$

$$N! = N \times (N-1)!$$

Dalam pemrograman dapat ditulis:

faktorial n = faktorial (n – 1) \* n

Dan

faktorial 0 = 1

Perhatikan bahwa ada dua definisi fungsi ini, yaitu untuk  $n = 0$  (yang hasilnya adalah 1) dan untuk  $n$  lebih dari nol (hasilnya adalah faktorial  $(n - 1)$  dikalikan dengan  $n$ ). Bagian yang tidak memanggil dirinya sendiri (dalam kasus  $n = 0$ ) disebut dengan basis, sedangkan bagian yang memanggil dirinya sendiri disebut sebagai bagian rekurens.

Secara lengkap fungsi faktorial dapat ditulis dalam pascal sebagai berikut:

```
Function faktorial(n: integer):integer;
begin
    if (n=0) then (*basis*)
        begin
            faktorial:= 1;
            end
    else (* rekurens *)
        begin
            faktorial:= n * faktorial ( n - 1);
            end;
    end;
```

### Penerapan Fungsi Rekursif

#### 1. FPB dari suatu Bilangan

FPB dari dua buah bilangan adalah suatu bilangan terbesar yang dapat membagi habis kedua bilangan tersebut (tanpa meninggalkan sisa). FPB biasanya dipelajari di matematika saat SD. Cara konvensionalnya biasanya adalah seperti pembahasan berikut.

Misalkan, ditanya: Berapa FPB dari 24 dan 30? Kalau zaman saya dulu, guru saya mengajarkan untuk menuliskan seluruh faktor yang ada dari kedua bilangan tersebut, kemudian mencari angka terbesar yang sama-sama dimiliki oleh kedua bilangan tersebut.

Faktor dari 24:

1, 2, 3, 4, [6], 8, 12 dan 24

Faktor dari 30:

1, 2, 3, 5, [6], 10, 15 dan 30

Seperti yang bisa pembaca lihat di atas, jadi berapa FPB dari 24 dan 30? Dari faktor yang telah diuraikan dari kedua bilangan tersebut, bilangan terbesar yang sama-sama dimiliki oleh kedua bilangan tersebut adalah 6. Berarti, 6 adalah FPB dari 24 dan 30.

Masalah di atas mungkin mudah diselesaikan jika angkanya relatif kecil, tetapi bagaimana kalau angkanya besar?

Kalau secara algoritmik, tentu akan sulit jika program yang dibuat harus mendata terlebih dahulu faktor-faktor dari masing-masing bilangan kemudian mencari angka terbesar yang sama-sama terdapat di daftar faktor dari kedua bilangan tersebut. Selain dibutuhkan variabel tambahan, tentu waktu komputasinya akan lebih lama. Karena itu butuh cara lain untuk menyelesaiakannya.

Deskripsi dari Algoritma Euclidean ini adalah sebagai berikut:

Misalkan terdapat dua buah bilangan yaitu m dan n, maka FPB dari kedua bilangan tersebut dapat dihitung dengan langkah berikut:

1. Bagi bilangan m dengan n dan anggap r adalah sisa bagi dari kedua bilangan tersebut.  
Nilai dari r akan memenuhi  $0 \leq r < n$ .
2. Jika nilai  $r = 0$ , maka algoritma selesai. n adalah solusinya. Tapi jika ternyata tidak, maka:
3. Set nilai m menjadi n, dan nilai n menjadi r, kemudian kembali ke langkah 1.

Misalkan, fpb adalah nama fungsi untuk menemukan FPB dari m dan n, berikut adalah contoh penggunaannya untuk kasus sebelumnya.

```
m = 24, n = 30
fpb (m, n):
24 / 30 = 0, r = 24
r tidak sama dengan 0, maka

m = n, n = r -> m = 30, n = 24
fpb (m, n):
30 / 24 = 1, r = 6
r tidak sama dengan 0, maka

m = n, n = r -> m = 24, n = 6
fpb (m, n):
24 / 6 = 0, r = 0
karena r = 0, maka hasilnya adalah n.
```

Jadi  $\text{fpb}(24, 30) = \text{fpb}(30, 24) = \text{fpb}(24, 6) = 6$

Algoritma tersebut, jika ditulis dalam Bahasa Pascal akan menjadi seperti berikut:

```
function fpb(m, n : integer) : integer;
var r : integer;
begin
    r := m mod n;

    while (r <> 0) do begin
        m := n;
        n := r;
        r := m mod n;
    end;

    fpb := n;
end;
```

Algoritma Euclid ini juga bisa ditulis dengan suatu model algoritma rekursif. Hal yang perlu diketahui adalah, pada kasus sebelumnya, jika pada fungsi  $\text{fpb}(m, n)$  nilai  $m$  dimasukkan dengan 0 maka akan secara otomatis nilai  $n$  akan dikembalikan. Begitu pula sebaliknya, jika pada fungsi  $\text{fpb}(m, n)$  nilai  $n$  dimasukkan dengan 0, maka secara otomatis nilai  $m$  akan dikembalikan. Dengan kata lain, FPB dari suatu bilangan dan 0 adalah bilangan itu sendiri. Berlandaskan pada gagasan tersebut, maka dapat ditarik suatu algoritma rekursif dengan definisi sebagai berikut:

1. Cek apakah ada di antara  $m$  atau  $n$  yang sama dengan 0? Jika ada hentikan algoritma dan hasil dari algoritma adalah bilangan yang bukan 0.
2. Jika baik  $m$  dan  $n$  tidak ada yang sama dengan 0, maka dengan mengasumsikan bahwa  $r$  adalah sisa bagi dari bilangan  $m$  dan  $n$ , maka  $\text{fpb}(m, n)$  itu sama dengan  $\text{fpb}(n, r)$ .

```
function fpb(m, n : integer) : integer;
begin
    if (m = 0) then
        fpb := n
    else if (n = 0) then
        fpb := m
    else begin
        fpb := fpb(n, m mod n);
    end;
end;
```

### 1.5.2.2. Menanya

Berdasarkan kegiatan mengamati, ada beberapa hal yang bisa diperhatikan dan dibahas lebih lanjut. Beberapa pertanyaan yang berkaitan dengan Pointer ke Variabel adalah:

1. Apakah yang akan terjadi bila fungsi rekursif tidak memiliki basis? Apa yang akan terjadi?
2. Apakah basis fungsi rekursif hanya bisa untuk satu nilai konstanta?
3. Apakah fungsi rekursif bisa memiliki basis lebih dari satu kasus atau kejadian?

Untuk menjawab pertanyaan-pertanyaan tersebut, siswa diharapkan bisa melakukan kegiatan mencoba berikut dengan bimbingan guru.

### 1.5.2.3. Mencoba

#### Percobaan 1

Untuk mengetahui jawaban dari pertanyaan pertama yaitu apa yang akan terjadi bila fungsi rekursif tidak memiliki basis, salin dan lengkapilah program berikut ke dalam FreePascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```
Program FaktorialTanpaBasis;
Function faktorial(n: integer):integer;
begin
    faktorial:= n * faktorial ( n - 1);
end;
var
    ..... : integer;
    {Mendeklarasikan variabel dengan tipe integer}
begin
    write('Masukan sembarang bilangan bulat positif :');
    {Meminta inputan dari user}
    Readln(.....);
    {Menyimpan inputan user ke dalam suatu variabel}
    Writeln('Nilai dari ',.....,'! = ',.....);
    {memanggil fungsi faktorial}
    readln;
end.
```

**Hasil Percobaan Program Faktorial Tanpa Basis:**

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

**Percobaan 2**

Untuk mengetahui jawaban dari pertanyaan kedua, Apakah basis fungsi rekursif hanya bisa untuk satu nilai konstanta?. Salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```
Program Perkalian;

Function kali(a,b:integer):longint;
begin
  if b>1 then
    kali := kali(a,b-1)+a
  else
    kali := a;
end; var
.....,..... : integer;
{Mendeklarasikan variabel dengan tipe integer}
begin
  write('Masukan bilangan pertama :');
  {Meminta inputan dari user}
```

```
Readln(.....);
{Menyimpan inputan user ke dalam suatu variabel}
write('Masukan bilangan kedua :');
{Meminta inputan dari user}

Readln(.....);
{Menyimpan inputan user ke dalam suatu variabel}

Writeln('Nilai dari ',.....,' x ',.....,' = ',.....);
{memanggil fungsi kali}
readln;
end.
```

Ingat!

Secara matematis, perkalian dua bilangan bulat positif a dengan b (ditulis ab atau a x b) pada hakekatnya merupakan penjumlahan dari a sebanyak b suku, yaitu a + a + a + .... + a sebanyak b suku. Misalnya  $2 \times 3$  dapat diartikan sebagai  $2 + 2 + 2 = 6$ , yaitu penjumlahan 2 sebanyak 3 suku. Dengan mengingat bahwa suatu bilangan bila dikalikan dengan angka 1 (satu) akan menghasilkan bilangan itu sendiri,

#### Hasil Percobaan Program Perkalian:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

#### Percobaan 3

Untuk mengetahui jawaban dari pertanyaan ketiga, Apakah fungsi rekursif bisa memiliki basis lebih dari satu kasus atau kejadian?. Salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan

kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```

Program Fibonacci;

Function fibo(n:integer): integer;
begin
    if (n=1) or (n=2) then
        fibo:=1
    else
        fibo(n) := fibo(n-1)+fibo(n-2);
end;
var
    .....: integer;
{Mendeklarasikan variabel dengan tipe integer}
begin
    write('Masukan sembarang bilangan bulat positif :');
    {Meminta inputan dari user}
    Readln(.....);
    {Menyimpan inputan user ke dalam suatu variabel}
    Writeln('Nilai dari bilangan Fibonacci ke', ....., ' = ', .....) ;
    {memanggil fungsi fibonacci}
    readln;
end.

```

Catatan:

Bilangan Fibonacci didefinisikan sebagai berikut

1 1 2 3 5 8 13 21 34 55 89 ...

dari barisan tersebut dapat dilihat bahwa bilangan ke-N ( $N > 2$ ) dalam barisan dapat dicari dari dua bilangan sebelumnya yang terdekat dengan bilangan N, yaitu bilangan ke-(N-1) dan bilangan ke-(N-2), sehingga dapat dirumuskan sebagai

$$\text{Fibonacci}(1) = 1 \quad (1)$$

$$\text{Fibonacci}(2) = 1 \quad (2)$$

$$\text{Fibonacci}(N) = \text{Fibonacci}(N-1) + \text{Fibonacci}(N-2) \quad (3)$$

Dengan persamaan (1) dan (2) adalah basis dan persamaan (3) adalah rekurensnya.

**Hasil Percobaan Program Fibonacci:**

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

**4.1.2.1. Mengasosiasi/ menalar**

Setelah siswa melakukan percobaan yang sesuai dengan pertanyaan yang ada, siswa diharapkan melakukan kegiatan menalar yang bisa dilakukan secara individu oleh masing-masing siswa atau bisa juga berkelompok. Beberapa hal yang bisa dijadikan arahan untuk menalar:

1. Perhatikan hasil kompilasi, apakah berhasil atau justru terjadi kesalahan.
2. Perhatikan urutan proses dari setiap program yang sudah dicoba.
3. Perhatikan peringatan kesalahan dan output dari program.
4. Cobalah membuat kesimpulan dari percobaan yang telah dilakukan.

**1.5.3. Rangkuman**

Dari percobaan-percobaan yang telah dilakukan, bisa ambil kesimpulan bahwa:

1. Fungsi Rekursif adalah fungsi yang memanggil dirinya sendiri jika selama kondisi pemanggilan dipenuhi.
2. Fungsi rekursif tidak akan terhenti jika tidak memiliki kondisi khusus atau basis
3. Fungsi rekursif dapat berjalan dengan baik meskipun basis bukan berupa nilai tetap
4. Fungsi rekursif dapat berjalan dengan baik meskipun basis lebih dari satu kondisi atau kasus

**1.5.4. Tugas**

1. Buatlah Program untuk menggambarkan konsep perpangkatan bilangan bulat menggunakan fungsi rekursif, dengan ketentuan sebagai berikut,

Perpangkatan dua bilangan bulat positif a dengan b (ditulis  $a^b$ ) pada hakikatnya merupakan perkalian dari a sebanyak b suku, yaitu  $a \times a \times a \times \dots \times a$  sebanyak b suku. Misalnya  $2^3$  dapat diartikan sebagai  $2 \times 2 \times 2 = 8$ , yaitu perkalian 2 sebanyak 3 suku

Susun laporan yang terdiri atas kode program, penjelasan program, dan output dari program.

2. . Perhatikan fungsi rekursif berikut:

$$\begin{aligned} F(x) &= 0 && x=0 \\ F(x) &= 2F(x-1)+2 && x \neq 0 \end{aligned}$$

Buatlah program pascal untuk menentukan nilai  $F(8)$ .

Susun laporan yang terdiri atas kode program, penjelasan program, dan output dari program.

#### 1.5.5. Uji Kompetensi

**Pilihlah salah satu jawaban yang paling tepat!**

1. Arti dari istilah Rekursif adalah....
  - a. suatu proses yang tidak bisa menunjuk dirinya sendiri.
  - b. suatu proses yang bisa menunjuk dirinya sendiri.
  - c. suatu proses yang tidak bisa memanggil dirinya sendiri.
  - d. suatu proses yang bisa memanggil dirinya sendiri
  - e. suatu proses yang bisa memanggil orang lain
2. Definisi dari Fungsi Rekursif adalah.....
  - a. Fungsi yang tidak memanggil dirinya sendiri selama kondisi pemanggilan dipenuhi
  - b. Fungsi yang menunjuk dirinya sendiri tanda ada kondisi pemanggilan yang dipenuhi
  - c. Fungsi yang menunjuk dirinya sendiri selama kondisi pemanggilan dipenuhi
  - d. Fungsi yang memanggil dirinya sendiri tanpa ada kondisi pemanggilan yang dipenuhi
  - e. Fungsi yang memanggil dirinya sendiri selama kondisi pemanggilan dipenuhi
3. Fungsi Rekursif disusun oleh dua bagian yaitu....
  - a. Basis dan Kondisi
  - b. Basis dan Logaritma
  - c. Basis dan Rekurens

- d. Basis dan Elemen  
 e. Basis dan Pokok
4. Perhatikan fungsi berikut:

$$F(x) = 0 \dots \dots \dots \quad x=0$$

$$F(x) = 2F(x-1)+2 \dots \dots \dots \quad x \neq 0$$

Bagian yang dinamakan basis adalah:

- a.  $F(x) = 0$
  - b.  $F(x) = 2F(x-1)+2$
  - c.  $x=0$
  - d.  $x \neq 0$
  - e.  $2F(x-1)+2$
5. Pernyataan yang tidak benar mengenai basis pada fungsi rekursif adalah ....
- a. Bagian yang berisi kasus yang terdefinisi secara eksplisit atau langsung
  - b. Bagian yang menghentikan fungsi rekursif
  - c. Bagian yang memberikan suatu nilai yang terdefinisi pada fungsi
  - d. Bagian yang menyebabkan pemanggilan diri fungsi
  - e. Kondisi yang menyebabkan pemanggilan dirinya berhenti
6. Berikut adalah Akibatnya Jika fungsi rekursif tidak memiliki basis....
- a. Proses rekursif tidak akan terhenti
  - b. Fungsi memiliki suatu nilai
  - c. Proses rekursif akan terhenti dengan sendirinya
  - d. Fungsi dapat berjalan dengan baik
  - e. Fungsi memberikan hasil sebagaimana mestinya
7. Perhatikan kode program berikut

function fpb(m, n : integer) : integer;	(1)
begin	(2)
if (m = 0) then	(3)
fpb := n	(4)
else if (n = 0) then	(5)
fpb := m	(6)
else begin	(7)
fpb := fpb(n, m mod n);	(8)
end;	(9)
end;	(10)

- Bagian blok (7) – (8) pada fungsi rekursif dinamakan....
- Basis
  - Rekurens
  - Rekursif
  - Fungsi
  - Kondisi
- Berikut adalah kelebihan dari fungsi rekursif....
    - Program berjalan lebih cepat
    - Kode Program menjadi lebih singkat
    - Bisa menangani setiap kasus dan masalah
    - Tidak memakan memori yang besar
    - Program menjadi lebih mudah untuk dibaca

**Jawablah Pertanyaan berikut dengan singkat dan jelas.**

- Jelaskan Pengertian dari Fungsi rekursif yang anda ketahui!
- Sebut dan jelaskan kelebihan dan kekurangan penggunaan fungsi rekursif?
- Apa yang akan terjadi jika dalam fungsi rekursif tidak memiliki basis?
- Apakah fungsi rekursif dapat diterapkan pada setiap kasus dan permasalahan?Apa alasan kamu?

## **1.6. Kegiatan Belajar 2 Pointer ke Fungsi dan Aplikasinya**

**Alokasi Waktu : 2 x 45 menit**

### **1.6.1. Tujuan Pembelajaran**

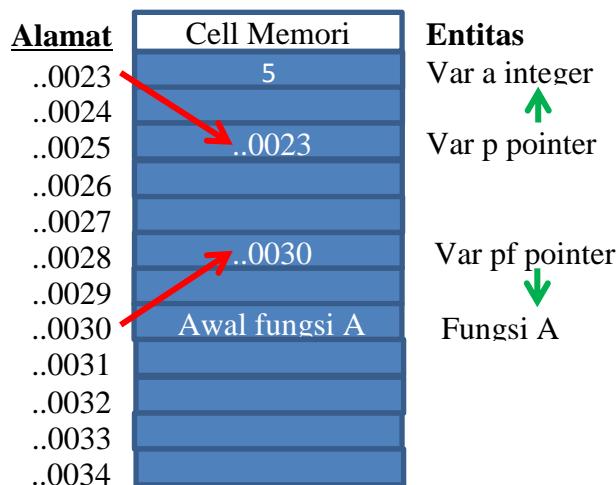
Tujuan pembelajaran pada Kegiatan Belajar 2 tentang Pointer ke fungsi adalah:

- Siswa dapat memahami mengarahkan pointer ke fungsi dan menggunakan dalam aplikasi

### **1.6.2. Aktivitas belajar siswa**

#### **1.6.2.1. Mengamati**

Pada pelajaran Pemrograman Dasar kelas XI Semester 1 sebelumnya telah dipelajari tentang pointer. Variabel bertipe pointer sendiri adalah suatu variabel yang bisa menunjuk variabel yang lain dengan cara menyimpan alamat memori dari variabel yang ditunjuk. Dalam pemrograman, setiap variabel, konstanta, prosedur dan fungsi akan menempati memori. Lalu bagaimana jika pointer kita arahkan supaya menunjuk ke suatu fungsi atau prosedur? Perhatikan skema berikut.

**Gambar 1. 4.** Representasi pointer pada memori**1.6.2.2. Menanya**

1. Bagaimana mengarahkan pointer supaya sebuah pointer bisa menuju suatu fungsi?
2. Bisakah pointer fungsi yang sudah ditentukan parameternya diarahkan ke fungsi dengan parameter yang tipenya berbeda?
3. Bisakah pointer menunjuk ke suatu prosedur?

**1.6.2.3. Mencoba****Percobaan 1**

Untuk mengetahui jawaban dari pertanyaan pertama, salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```

program PointFunct1;
uses crt;

//definisikan tipe point adalah pinter ke suatu fungsi
type point=Function(x:word):word;

//deklarasikan variabel a bertipe point
var a:point;

function ASD(x:word):word;
begin
  x:=x*x;
  ASD:=x;
end;

//Program Utama
begin
  //arahkan a supaya menunjuk fungsi ASD
  a:=@ASD;
  // panggil fungsi ASD dengan menggunakan pointer a

```

```
writeln(a(6));
readln;
end.
```

**Hasil Percobaan Program PointFunct1:**

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

**Percobaan 2**

Untuk mengetahui jawaban dari pertanyaan kedua, salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```
program PointFunct2;

uses crt;

type point=Function(x:word):word;
      pointProc=Procedure;

var a:point;
    b:pointProc;
    nama:PChar='Dhidhi Pembudi';

function ASD(x:word):word;
begin
    x:=x*x;
    ASD:=x;
end;

function BCD(y:byte):byte;
begin
    BCD:=y*y*y;
end;
```

```

begin
    a:=@ASD;
    writeln(a(6));
    a:=@BCD;
    writeln(a(3));
    readln;

end.

```

**Hasil Percobaan Program PointFunct2:**

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Salin pesan asli kompilasi disini:

.....  
.....  
.....

Letak kesalahan (baris, kolom)	Terjadi karena

Bagaimana cara memperbaiki kesalahan tersebut?

**Percobaan 3**

Untuk mengetahui jawaban dari pertanyaan ketiga, salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```

program pointFunct3;
uses crt;
type point=Function(x:word):word;
    pointProc=Procedure;
var a:point;
    b:pointProc;
    nama:PChar='Dhidhi Pembudi';
function ASD(x:word):word;

```

```

begin
    x:=x*x;
    ASD:=x;
end;

function BCD(y:word) :word;
begin
    BCD:=y*y*y;
end;
procedure FGH;
begin
    writeln(ASD(2)-BCD(3));
end;

function Power3(x:word): integer; stdcall; external 'subsku'
name 'Pangkat3';

begin
    a:=@ASD;
    writeln(a(6));
    a:=@BCD;
    writeln(a(3));
    b:=@FGH;
    b;
    writeln(Power3(3));
    readln;
end.

```

#### Hasil Percobaan Program PointFunct3:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

#### 1.6.2.4. Mengasosiasi/ menalar

Setelah siswa melakukan percobaan yang sesuai dengan pertanyaan yang ada, siswa diharapkan melakukan kegiatan menalar yang bisa dilakukan secara individu oleh masing-masing siswa atau bisa juga berkelompok. Beberapa hal yang bisa dijadikan arahan untuk menalar:

1. Perhatikan hasil kompilasi, apakah berhasil atau justru terjadi kesalahan.

2. Perhatikan urutan proses dari setiap program yang sudah dicoba.
3. Perhatikan letak/posisi kesalahan, coba pikirkan mengapa terjadi kesalahan pada posisi tersebut.
4. Cobalah membuat kesimpulan dari percobaan yang telah dilakukan.

#### 1.6.3. Rangkuman

2. Pointer ke fungsi adalah sebuah pointer yang digunakan untuk menunjuk suatu fungsi yang sesuai, sehingga pemanggilan fungsi tidak harus dengan menyebut nama fungsi secara langsung tetapi cukup menggunakan pointernya. Meski sebuah fungsi sudah ditunjuk oleh suatu pointer tetapi tetap diperkenankan memanggil fungsi dengan nama asli fungsi tersebut.
3. Untuk mengarahkan pointer supaya menunjuk ke sebuah fungsi, cukup gunakan operator @. Langkah-langkahnya adalah:
  - i. Deklarasikan variabel bertipe pointer ke fungsi, atau bisa didahului dengan mendefinisikan tipe data sendiri yaitu tipe data pointer ke fungsi.
  - ii. Arahkan varibel pointer ke sebuah fungsi yang sesuai dengan operator @.
  - iii. Gunakan pointer ke fungsi untuk memanggil atau menggunakan fungsi tersebut.
4. Sebuah pointer fungsi yang sudah ditentukan tipe parameternya tidak bisa digunakan untuk menunjuk fungsi lain yang tipe parameternya berbeda. Nama variabel dari parameter fungsi boleh berbeda tetapi tipe harus sama.
5. Sebuah pointer juga bisa menunjuk suatu prosedur dengan cara yang serupa dengan bagaimana sebuah pointer menunjuk suatu fungsi.

#### 1.6.4. Tugas

1. Coba pikirkan bagaimana sebuah pointer secara bergantian bisa menunjuk ke fungsi-fungsi lain yang tipe parameternya berbeda-beda. Buatlah dalam contoh program.
2. Coba pikirkan bagaimana sebuah pointer bisa menunjuk prosedur yang tipe parameternya sudah ditentukan. Buatlah dalam contoh program.
3. Modifikasilah fungsi dan prosedur buatanmu dari pelajaran-pelajaran sebelumnya agar bisa digunakan dengan pointer.

## 1.7. Kegiatan Belajar 7. Modularisasi Fungsi

**Alokasi Waktu : 3 x 45 menit**

### 1.7.1. Tujuan Pembelajaran

Tujuan pembelajaran pada Kegiatan Belajar 3 tentang Pemanfaatan Pointer adalah:

1. Siswa dapat memahami konsep modularisasi fungsi.
2. Siswa dapat memahami teknik modularisasi menggunakan file library
3. Siswa dapat memahami teknik modularisasi menggunakan unit

### 1.7.2. Aktivitas belajar siswa

Untuk mencapai suatu tujuan besar, maka tujuan tersebut harus dibagi-bagi menjadi tujuan kecil sampai tujuan kecil itu merupakan tujuan yang dapat dicapai berdasarkan kondisi dan potensi yang dimiliki saat itu (Al-Khuwarizmi). Ucapan Al-Khuwarizmi di atas sangat mengena dalam kegiatan memprogram. Program yang besar lebih sulit dimengerti (dibaca) dan lebih sulit lagi dalam melakukan pelacakan kesalahan jika ada). Oleh karena itu, program sebaiknya dipecah menjadi beberapa subprogram yang lebih kecil. Setiap subprogram melakukan komputasi yang spesifik. Subprogram yang baik adalah subprogram yang independen dari program utama sehingga programnya dapat dirancang tanpa mempertimbangkan konteks di mana ia digunakan. Dengan kata lain, pemrogram tidak perlu · mempermasalahkan bagaimana subprogram tersebut dilakukan, tetapi cukup memikirkan apa yang ia lakukan. Subprogram yang bagus menyembunyikan detail operasi dari bagian program yang tidak perlu tahu tentang subprogram tersebut. Teknik pemecahan program menjadi sejumlah subprogram dinamakan teknik pemrograman modular. Sehingga dalam kegiatan belajar kali ini akan di bahas pembuatan fungsi yang terpisah dari program utamanya. Hal ini dalam pascal dapat menggunakan file library dan unit.

#### 1.7.2.1. Mengamati

##### Definisi File Library

File dynamic library adalah file pustaka yang di dalamnya terdapat prosedur-prosedur ataupun fungsi-fungsi yang berguna untuk menyimpan proses untuk kemudian dipanggil hasilnya oleh program utama. File ini sangat berguna ketika kita membuat program yang sangat kompleks, sehingga resource yang kita gunakan bisa lebih hemat, terutama resource memory.

##### Membuat File library

Dalam menyusun libarary, struktur yang harus kita ikuti adalah sebagai berikut :

**Library namalibrary;**

{nama pustaka dan nama file sumber (.pas) harus sama, karena

```

file dynamic bersifat case sensitif}

Function namafungsi;export
{pengenal export (tanpa huruf 's') merupakan pengenal bahwa
hasil dari proses bisa dipanggil oleh program utama}

Procedure namaprocedure;export;

Exports namafungsi name 'namafungsi';
{export dengan huruf s, dan name menunjukan nama yang
dipakai untuk pemanggilan oleh program utama}
Exports namaprocedur name 'namaprocedur';

Begin
End.

```

### Kompilasi Unit

Setelah library dikompilasi (Compile / Alt+F9) akan menghasilkan file dengan extention .dll yang siap dipanggil oleh program utama.

Contoh 1. Library fungsi dan prosedur penjumlahan dan pengurangan  
**Library fungsi;**

**Function hasil(var a,b:integer):real;export;**

Begin

Hasil:= a+b

End;

**Procedure p\_hasil(a,b:integer; var c:real);export;**

Begin

c:= a-b

End;

**Exports hasil name 'hasil';**

**Exports p\_hasil name 'p\_hasil';**

**Begin**

**End.**

Setelah kode di atas kita tulis dalam pascal, disimpan dengan nama fungsi.pas kemudian kita compile, maka akan terbentuk file fungsi.dll. file inilah yang bisa dipanggil program utama (.exe).

### Pemanggilan File library

Untuk memanggil fungsi atau prosedur yang telah kita buat, struktur yang harus kita perhatikan adalah sebagai berikut:

```
Function namafungsi;external 'namadirektori/nama_file_dll' name  
'namafungsi';
```

Dari kode diatas, untuk memanggil nama fungsi, kita letakan pengenal external kemudian kita tunjukan nama beserta letak file dll kita. Begitu juga dengan memanggil prosedur yang kita buat.

### Contoh 2. Memanggil File Library

```
Program Pejumlahan;  
  
Uses crt;  
  
Function hasil(var a,b:integer):real;external './fungsi.dll' name  
'hasil';  
  
Procedure p_hasil(a,b:integer; var  
c:real);external './fungsi.dll' name 'p_hasil';  
  
Var  
    j,k:integer;  
    H:real;  
  
Begin  
    Clrscr;  
    write('nilai a = ');readln(j);  
    write('nilai b = ');readln(k);  
    write('fungsi a+b =',hasil(j,k):2:0);  
    P_hasil(k,j,h);  
    Readln;  
End.
```

Lakukan file library (.dll) dan program utama dalam satu folder. Kemudian compile program utama.

### Definisi Unit

Unit adalah suatu modul program yang terpisah dan digunakan untuk menyimpan proses-proses tertentu yang berkaitan. unit dibuat untuk program yang lumayan besar sehingga program dapat lebih bersifat modular dan mudah untuk dibaca.

### Struktur Unit

Sebelum membuat unit, kita harus mengetahui dulu struktur dari unit itu sendiri. Unit memiliki struktur tertentu dalam proses pembuatannya. Umumnya unit terbagi jadi tiga bagian, yaitu interface, implementation dan initialization, bentuk umumnya bisa seperti ini:

```

unit (NamaUnit);

interface

  uses      (daftar unit);
  const     (konstanta);
  var       (nama variabel);
  procedure (prototipe prosedur);
  function  (prototipe fungsi);

implementation

  uses      (daftar unit);
  const     (konstanta);
  var       (nama variabel)';
  procedure (implementasi prosedur);
  function  (implementasi fungsi);

begin
  {initialization};
  (statement);
end.

```

Bagian Interface, pada bagian ini isinya adalah daftar unit yang akan digunakan, deklarasi konstanta maupun variabel bila ada dan prototipe prosedur dan fungsi. jadi pada bagian ini, prosedur ataupun fungsi hanya dikenalkan saja dan implementasinya baru akan ditulis di dalam bagian implementation. Contoh prototipe prosedur dan fungsi seperti ini

```

function tambah(x, y:integer):integer;
function kali(x, y:integer):integer;
procedure tukar(var x, y:integer);

```

Hal yang perlu diperhatikan adalah, bagian ini (interface) harus ada di setiap pembentukan setiap unit. Bagian Implementation Sama dengan bagian interface, implementation juga dapat berisi daftar unit yang akan digunakan, deklarasi konstanta maupun variabel(bila ada) dan implementasi dari prosedur maupun fungsi yang sebelumnya udah dideklarasikan pada bagian interface dengan menuliskan prototipe, contohnya sebagai berikut:

```

function tambah(x, y:integer):integer;
begin
  tambah:=x+y;
end;

```

```

function kali(x, y:integer):integer;
begin
  kali:=x*y;
end;

procedure tukar(var x, y:integer);
var
  z:integer;
begin
  z:=x;
  x:=y;
  y:=z;
end;

```

bagian implementation ini juga harus ada dalam setiap pembentukan unit. Bagian Initialization Kalau bagian ini sifatnya opsional. Jadi bisa dituliskan, bisa juga tidak. fungsi dari bagian initialization ini sendiri adalah untuk melakukan inisialisasi nilai terhadap variabel global yang ada di dalam unit. Ini contoh pengisian inisialisasi pada bagian ini:

```

Begin
  x:=0;
  y:=0;
  nama:=' ';
  ...
end.

```

Catatan: bagian initialization ini berada di dalam blok begin-end.

### Kompilasi Unit

Unit dapat dikompilasi seperti halnya program biasa (Compile / Alt+F9). Hasil dari kompilasi tersebut menghasilkan extention .PPU

### Pemanggilan Unit

Untuk memanggil unit yang telah kita buat digunakan perintah sebagai berikut:

```
Uses Namaunit;
```

Sehingga semua fungsi dan prosedur pada unit tersebut dapat dipanggil dan digunakan dalam program utama.

## Contoh 1. Unit Contoh

```
unit contoh;

interface

    function tambah(x, y:integer):integer;
    function kali(x, y:integer):integer;
    procedure tukar(var x, y:integer);

implementation

function tambah(x, y:integer):integer;
begin
    tambah:=x+y;
end;

function kali(x, y:integer):integer;
begin
    kali:=x*y;
end;

procedure tukar(var x, y:integer);
var
    z:integer;
begin
    z:=x;
    x:=y;
    y:=z;
end;

{bagian initialization ini tidak diisi}
end.
```

simpan file tersebut dengan nama contoh.pas. yang harus diingat adalah nama file harus sama dengan nama unit. Kemudian, cukup kompilasi unit tersebut, bukan dijalankan. Sekarang, kita akan buat program yang akan menggunakan unit yang telah kita buat di atas.

## Contoh 2. Program Coba Unit

```
program CobaUnit;

uses
  crt, contoh; {menggunakan unit crt dan contoh}

var
  hasiltambah, hasilkali:integer;
begin
  clrscr;
  hasiltambah:=tambah(5, 10);
  hasilkali :=kali(5,10);
  writeln('5 + 10 = ',hasiltambah);
  writeln('5 x 10 = ',hasilkali);
  readln;
end.
```

### 1.7.2.2. Menanya

1. Apakah nama library pada program harus sama dengan nama file library ketika disimpan?Apa yang akan terjadi jika berbeda?
2. Apakah dalam satu library diperbolehkan jika hanya terdapat satu fungsi?
3. Apakah nama unit pada program harus sama dengan nama file unit ketika disimpan?Apa yang akan terjadi jika berbeda?
4. Apakah dalam satu library diperbolehkan jika hanya terdapat satu fungsi?

### 1.7.2.3. Mencoba

#### Percobaan 1

Untuk mengetahui jawaban dari pertanyaan pertama yaitu Apakah nama library pada program harus sama dengan nama file library ketika disimpan?Apa yang akan terjadi jika berbeda?. Salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi.

```

Library kalipangkat;
Function hasilkali(var .....,:integer):real;export;
Begin
.....:= a*b
{Menyimpan nilai hasil kali ke fungsi}
End;
Function hasilkuadrat(var .....:integer):real;export;
Begin
hasilkali:= a*a
End;

Exports hasilkali name ‘hasilkali’;
Exports hasilkuadrat name ‘hasilkuadrat’;

Begin
End

```

Simpan kode program dengan nama perkalian.pas kemudian compile, kemudian buat program pemanggil library tersebut:

```

Program KaliDanPangkat;
Uses .....
{Unit untuk membersihkan layar}

Function .....(var a,b:integer):real;external'./.....dll'name
'.....';
{Memanggil fungsi perkalian}

Function .....(.....:integer; var
c:real);external'./.....dll'name '.....';
{Memanggil fungsi pangkat dua}

Var
.....,.....:integer;
{Mendeklarasikan variabel bertipe integer}

Begin
Clrscr;
write('nilai a = ');
.....(j);
{Menyimpan inputan user ke dalam variabel}
write('nilai b = ');
.....(k);
{Menyimpan inputan user ke dalam variabel}
write('fungsi a*b = ',.....(j,k):2:0);

```

```
{Memanggil fungsi perkalian}
write('fungsi a^2 =', .....(.....):2:0);
{Memanggil fungsi pangkat dua}
write('fungsi b^2 =', .....(.....):2:0);
{Memanggil fungsi pangkat dua}
Readln;
End.
```

### Hasil Percobaan Program KaliDanPangkat:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

### Percobaan 2

Untuk mengetahui jawaban dari pertanyaan kedua yaitu Apakah dalam satu library diperbolehkan jika hanya terdapat satu fungsi? Apa yang akan terjadi jika berbeda?. Salin dan lengkapilah program berikut ke dalam FreePascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi.

```
Library kalipangkat;
Function hasilkali(var .....,:integer):real;export;
Begin
.....:= a*b
{Menyimpan nilai hasil kali ke fungsi}
End;
Exports hasilkali name 'hasilkali';
Begin
End
```

Simpan kode program dengan nama perkalian.pas kemudian compile, kemudian buat program pemanggil library tersebut:

```
Program Kali;
Uses .....
{Unit untuk membersihkan layar}
Function .....(var a,b:integer):real;external'./.....dll'name
'.....';
```

```

{Memanggil fungsi perkalian}

Var
....., .....:integer;
{Mendeklarasikan variabel bertipe integer}

Begin
Clrscr;
write('nilai a = ');
.....(j);
{Menyimpan inputan user ke dalam variabel}
write('nilai b = ');
.....(k);
{Menyimpan inputan user ke dalam variabel}
write('fungsi a*b = ', .....(j,k):2:0);
Readln;
End.

```

### Hasil Percobaan Program Kali:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

### Percobaan 3

Untuk mengetahui jawaban dari pertanyaan ketiga yaitu Apakah nama unit pada program harus sama dengan nama file unit ketika disimpan? Apa yang akan terjadi jika berbeda? Salin dan lengkapilah program berikut ke dalam Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```

unit kuadratkubik;
interface
  function kuadrat(x:integer):integer;

```

```

function kubik(x:integer):integer;

implementation

function kuadrat(x:integer):integer;
begin
.....:=x*x;
{Memberikan nilai balikan pada fungsi}
end;

function kubik(x:integer):integer;
begin
.....:=x*x*x;
{Memberikan nilai balikan pada fungsi}
end;
{bagian initialization ini tidak diisi}
end.

```

Simpan kode program dengan nama kuadrat.pas kemudian compile.

#### Hasil Percobaan Unit kuadratkubik:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Salin pesan asli kompilasi disini:

.....  
.....  
.....

Letak kesalahan (baris, kolom)	Terjadi karena

#### Percobaan 4

Untuk mengetahui jawaban dari pertanyaan keempat yaitu Apakah dalam satu unit diperbolehkan jika hanya terdapat satu fungsi? Salin dan lengkapilah program berikut ke dalam

Freepascal, kemudian lakukan kompilasi (Compile / Alt+F9). Jika kompilasi menghasilkan pesan kesalahan maka perhatikan kesalahan yang terjadi. Jika kompilasi berhasil lanjutkan dengan menjalankan program (Ctrl+F9) dan amati hasilnya.

```
unit kuadrat;

interface

function hasilkuadrat(x:integer):integer;

implementation

function hasilkuadrat(x:integer):integer;
begin
.....:=x*x;
{Memberikan nilai balikan pada fungsi}
end;

{bagian initialization ini tidak diisi}
end.
```

Simpan kode program dengan nama kuadrat.pas kemudian compile. kemudian buat program pemanggil unit tersebut:

```
program MencariKuadrat;

uses
  crt, kuadrat; {menggunakan unit crt dan contoh}

var
  angka,hasil:integer;
begin
  clrscr;
  write('Masukan Sembarang bilangan bulat : ');
  readln(angka);
  hasil:=hasilkuadrat(.....);
  {Memanggil fungsi hasil kuadrat}
  writeln('Nilai dari ',angka,'^2 = ',.....);
```

```
{Menampilkan nilai fungsi hasil kuadrat}  
readln;  
end.
```

#### Hasil Percobaan Program MencariKuadrat:

Hasil Kompilasi (beri tanda silang pada bagian yang sesuai)

- Berhasil, tanpa kesalahan
- Tidak berhasil, ada kesalahan

Output Program:

#### 1.7.2.4. Mengasosiasi/ menalar

Setelah siswa melakukan percobaan yang sesuai dengan pertanyaan yang ada, siswa diharapkan melakukan kegiatan menalar yang bisa dilakukan secara individu oleh masing-masing siswa atau bisa juga berkelompok. Beberapa hal yang bisa dijadikan arahan untuk menalar:

1. Perhatikan hasil kompilasi, apakah berhasil atau justru terjadi kesalahan.
2. Perhatikan urutan proses dari setiap program yang sudah dicoba.
3. Perhatikan letak/posisi kesalahan, coba pikirkan mengapa terjadi kesalahan pada posisi tersebut.
4. Cobalah membuat kesimpulan dari percobaan yang telah dilakukan.

#### 1.7.3. Rangkuman

1. File dynamic library adalah file pustaka yang di dalamnya terdapat prosedur-prosedur ataupun fungsi-fungsi yang berguna untuk menyimpan proses untuk kemudian dipanggil hasilnya oleh program utama
2. Penamaan library boleh berbeda dengan nama file library tersebut
3. Unit adalah suatu modul program yang terpisah dan digunakan untuk menyimpan proses-proses tertentu yang berkaitan
4. Penamaan unit harus sama dengan nama file unit tersebut.

#### 1.7.4. Tugas

Buatlah library dan unit hasil modifikasi fungsi dan prosedur yang sudah dipelajari pada Kegiatan Belajar Prosedur dan Fungsi.

#### 1.7.5. Uji Kompetensi

**Pilihlah salah satu jawaban yang paling tepat!**

1. Arti dari istilah Teknik Modularisasi dalam pemrograman adalah....
  - a. Teknik pengurutan program menjadi sejumlah subprogram
  - b. Teknik penyederhanaan program sehingga program lebih cepat
  - c. Teknik Pencarian program menjadi sejumlah subprogram
  - d. Teknik penggabungan program menjadi suatu subprogram
  - e. Teknik pemecahan program menjadi sejumlah subprogram
2. Di bawah ini yang bukan merupakan kelebihan modularisasi dalam pemrograman adalah....
  - a. Fungsi tidak dapat dipanggil berulang kali,
  - b. Penulisan program jadi lebih lambat
  - c. Penulisan kode program jadi lebih panjang
  - d. Lebih sulit menemukan kesalahan program
  - e. Lebih mudah menemukan kesalahan dalam program
3. Manakah pernyataan berikut yang benar mengenai file library adalah....
  - a. Tidak dapat menyimpan prosedur atau fungsi lebih dari satu
  - b. Nama library harus sama dengan nama file library
  - c. Hasil compile dari library berekstensi .exe
  - d. Hasil compile dari library berekstensi .ppu
  - e. Nama library boleh berbeda dengan nama file library
4. Manakah pernyataan berikut yang tidak benar mengenai file unit adalah....
  - a. Dapat menyimpan prosedur atau fungsi lebih dari satu
  - b. Nama library harus sama dengan nama file library
  - c. Hasil compile dari library berekstensi .exe
  - d. Hasil compile dari library berekstensi .ppu
  - e. Untuk memanggilnya menggunakan perintah uses

# BAB II

## *PENCARIAN DAN PENGURUTAN DATA*

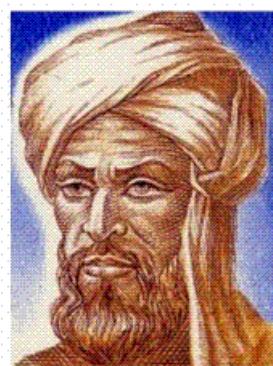
### Kompetensi Dasar:

- Menerapkan logika pencarian data
- Menerapkan logika pengurutan data
- Menyajikan logika pencarian data pada aplikasi bisnis
- Menyajikan logika pengurutan data pada aplikasi bisnis

## BAB II

### PENCARIAN DAN PENGURUTAN DATA

Berbicara mengenai pencarian dan mengurutkan data, pasti ada langkah-langkah yang harus dilakukan. Mengenai langkah-langkah tersebut, algoritma adalah suatu urutan langkah-langkah untuk memecahkan atau menyelesaikan suatu masalah. Selain itu, ada yang menyebutkan bahwa algoritma merupakan deretan langkah komputasi yang mentransformasikan masukan menjadi keluaran. Algoritma berasal dari kata *algorism* yang berasal dari nama penulis buku, yakni Abu Ja'far Muhammad Ibnu Musa Al-Khwarizmi yang berasal dari Uzbekistan. Orang Barat menyebut Al-Khwarizmi dengan Algorism. Pada saat itu, Al-Khwarizmi menulis buku dengan judul *Al Jabar wal-Muqabala* yang artinya "Buku Pemugaran dan Pengurangan" (*The book of Restoration and Reduction*). Dari judul buku tersebut, kita juga memperoleh kata "aljabar" atau biasa dikenal dengan *algebra*. Awalnya, algoritma merupakan istilah yang merujuk kepada aturan-aturan aritmetis yang berguna untuk menyelesaikan persoalan dengan menggunakan bilangan *numeric* Arab.



**Gambar 4.** Sketsa wajah al-Khawarizmi (biskerja.com)

#### 2.1. Kegiatan Belajar 1. Pencarian Data Dengan Algoritma Linier

**Alokasi Waktu : 2x45 Menit**

##### 2.1.1. Tujuan Pembelajaran

Tujuan pembelajaran pada Kegiatan Belajar 1 tentang pencarian dan pengurutan data adalah:

1. Peserta didik mampu memahami algoritma-algoritma dasar dalam pencarian data.
2. Peserta didik mampu memahami algoritma pencarian linier.
3. Peserta didik mampu membuat algoritma linier pada data array terurut.
4. Peserta didik mampu membuat algoritma linier pada data array sentinel.

## 2.1.2. Aktivitas Belajar Siswa

### 2.1.2.1. Mengamati



**Gambar 2.1.** Belanja di supermarket (<http://tips7itu.blogspot.com>)

Di dalam sebuah supermarket berisi berbagai macam barang yang dijual. Setiap barang diletakkan sesuai dengan kumpulan dan jenis-jenis barangnya. Setiap pembeli diharuskan mengambil dan mencari sendiri barang yang akan dibeli. Bagaimana proses pembeli membeli sabun tersebut? Jelaskan mulai dari proses mencari hingga mendapatkan barangnya!

Proses tersebut dapat dilakukan dengan langkah-langkah sebagai berikut:

Langkah 0 → kita berada di supermarket.

Langkah 1 → mencari lokasi kumpulan barang tersebut (dalam konteks ini, kita mencari lokasi kumpulan sabun).

Langkah 2 → mencari sabun mandi di lokasi kumpulan sabun.

Langkah 3 → jika sabun mandi yang dicari tersedia di supermarket, maka kita mendapatkan sabun mandi yang dicari (mendapat hasil) .

Langkah 4 → menuju kasir dan membayar.

Pencarian (*searching*) merupakan proses fundamental dalam pengolahan suatu data.

Proses pencarian adalah menemukan nilai (data) tertentu di dalam sekumpulan data yang bertipe sama (baik bertipe dasar atau bertipe bentukan). Sebagai contoh adalah untuk mengubah (*update*) data tertentu, langkah pertama yang harus dilakukan adalah mencari keberadaan data tersebut di dalam kumpulannya. Jika data yang dicari ditemukan, maka data tersebut dapat diubah nilainya dengan data yang baru. Sama halnya dengan kita membeli suatu barang di supermarket yang mengharuskan kita untuk mengambil sendiri barang yang akan kita beli, langkah pertama yang kita lakukan adalah mencari keberadaan barang tersebut di dalam kumpulannya. Misalnya kita akan membeli sabun mandi. Kita harus mencari sabun mandi di dalam kumpulannya (rak-rak tempat sabun mandi, karena di supermarket barang-barang yang sejenis ditempat dalam satu tempat). Kemudian baru kita cari sabun mandi yang akan kita beli. Setelah sabun didapat, baru kita ke kasir dan membayarnya.

Apakah aktivitas awal yang harus dilakukan adalah sama dalam proses penambahan (*insert*) data baru seperti halnya pelayan supermarket menambahkan barang dalam supermarket?

Dalam buku ini akan dibahas algoritma pencarian data dalam array yang hanya akan membahas algoritma pencarian yang paling sederhana yaitu algoritma pencarian linier atau pencarian beruntun atau *sequential search*.

### Spesifikasi Masalah Pencarian

Pertama-tama kita menspesifikasikan masalah pencarian di dalam array. Dalam bab 6 ini kita mendefinisikan permasalahan pencarian secara umum sebagai berikut: *diberikan array A yang sudah terdefinisi elemen-elemennya, dan x adalah elemen yang bertipe sama dengan elemen array A. Carilah x di dalam array A!*

Hasil atau keluaran dari persoalan atau permasalahan pencarian dapat bermacam-macam, tergantung pada spesifikasi rinci dari persoalan atau permasalahan yang dihadapi tersebut, misalnya:

- Pencarian yang hanya untuk memeriksa keberadaan x. Hasil atau keluaran yang diinginkan misalnya hanya pesan bahwa x ditemukan atau tidak ditemukan di dalam array.

Contoh:

```
Write ('ditemukan!') atau
Write ('tidak ditemukan!')
```

- Hasil pencarian adalah indeks elemen array. Jika x ditemukan, maka indeks elemen array tempat x berada diisikan ke dalam *idx*. Jika x tidak terdapat di dalam array (misal array A), maka *idx* diisi dengan harga khusus, misal -1.

Contoh: perhatikan array di bawah ini

A	22	32	4	3	10	36	99	56	91	4
	1	2	3	4	5	6	7	8	9	10

Misalkan  $x = 99$ , maka  $idx = 7$ , dan bila  $x = 50$ , maka  $idx = -1$ .

- Hasil pencarian adalah sebuah nilai *boolean* yang menyatakan status hasil pencarian. Jika x ditemukan, maka sebuah peubah bertipe *boolean*, misalnya “ketemu”, diisi dengan nilai true, sebaliknya “ketemu” diisi dengan nilai false. Hasil pencarian ini selanjutnya disimpulkan pada bagian pemanggil prosedur.

Contoh: Perhatikan array A di atas:

Misalkan  $x = 99$ , maka  $\text{ketemu} = \text{true}$ , dan bila  $x = 50$ , maka  $\text{ketemu} = \text{false}$ .

Untuk kedua macam keluaran b dan c di atas, kita mengkonsultasi hasil pencarian setelah proses pencarian selesai dilakukan, tergantung pada kebutuhan. Misalnya menampilkan pesan x ditemukan (atau tidak ditemukan) atau memanipulasi nilai x.

### Algoritma Pencarian linier

Algoritma pencarian yang paling sederhana, yaitu metode pencarian linier (pencarian lurus). Nama lain algoritma linier adalah algoritma pencarian beruntun (*sequential search*). Pada

dasarnya, algoritma pencarian linier adalah proses membandingkan setiap elemen array satu persatu secara beruntun, mulai dari elemen pertama sampai elemen yang dicari ditemukan atau seluruh elemen sudah diperiksa.

Misalkan, pada sebuah kasus anda ingin menelepon teman, tapi kebetulan lupa dengan nomor teleponnya. Maka anda pasti membuka buku telepon untuk mencari nomor telepon yang anda inginkan tersebut. Di sisi lain, buku telepon anda tidak urut sama sekali, baik dari segi nama yang sesuai urutan huruf abjad, maupun dari nomor telepon. Berarti anda harus mencari satu persatu, halaman demi halaman nomor telepon tersebut sampai akhirnya ditemukan nomor telepon tersebut. Begitulah kira-kita algoritma pencarian linier.

Algoritma pencarian yang paling sederhana adalah metode pencarian linier. Seperti yang dibahas pada kegiatan sebelumnya, kita akan melakukan pencarian pada sebuah array yang sudah terdefinisi elemen-elemennya, dan  $x$  adalah elemen yang bertipe sama dengan elemen array A carilah  $x$  di dalam array A.

Misalkan array A di bawah ini dengan  $n = 7$  elemen:

A	22	32	4	9	10	36	46
	1	2	3	4	5	6	7

Misalkan nilai yang kita cari adalah  $x = 36$ .

Elemen yang dibandingkan berturut-turut adalah 22, 32, 4, 9, 10, 36 (ditemukan)

Indeks array yang dikembalikan adalah 6

Misalkan nilai yang kita cari adalah  $x = 22$ .

Elemen yang dibandingkan berturut-turut adalah 22 (ditemukan)

Indeks array yang dikembalikan adalah 1

Misalkan nilai yang kita cari adalah  $x = 100$ .

Elemen yang dibandingkan berturut-turut adalah 22, 32, 4, 9, 10, 36, 46 (tidak ditemukan)

Indeks array yang dikembalikan adalah -1

Pada algoritma pencarian linier pada array A, setiap elemen array A dibandingkan dengan  $x$  mulai dari elemen pertama hingga  $A[1]$ . Aksi pembandingan dilakukan selama indeks array i belum melebihi  $n$  dan  $A[i]$  tidak sama dengan  $x$ . Aksi pembandingan dihentikan bila  $A[i] = x$  atau  $i = n$ . Elemen terakhir,  $A[n]$ , diperiksa secara khusus. Keluaran yang dihasilkan oleh prosedur pencarian adalah sebuah boolean (misal nama peubahnya *ketemu*) yang bernilai *true* jika  $x$  ditemukan, atau bernilai *false* jika  $x$  tidak ditemukan.

Algoritma bisa ditulis sebagai berikut:

(i) Prosedur pencarian beruntun:

```
Procedure liniersearch1 (input A : arrayint, input n :
```

```

integer, input x : integer, output ketemu : boolean)

{mencari keberadaan nilai x di dalam array A[1..n].}

{k.awal: x dan array [1..n] sudah terdefinisi nilainya.}

{k.akhir: ketemu nilai true jika x ditemukan. Jika x
tidak ditemukan, ketemu bernilai false.}

```

Deklarasikan

```
I: integer {pencatat indeks array}
```

Algoritma:

```

i ← 1

while (i < n) and (A[i] ≠ x) do

    i ← i + 1

endwhile

{i = n or A[i] = x}

if A[i] = x then      {x ditemukan}

    Ketemu ← true

else

    ketemu ← false   {x tidak ada di dalam array A}

endif

```

## (ii) penulisan freepascal algoritma pencarian linier

```

(*deklarasi*)

const Nmaks = 100; {jumlah maksimum elemen array}

type arrayint = array [1..Nmaks] of integer;

```

```

Procedure liniersearch1 (input L : arrayint; input n :
integer; input x : integer; var idx : integer;

{mencari keberadaan nilai x di dalam array A[1..n].}

{k.awal: x dan elemen -elemen array [1..n] sudah
terdefinisi nilainya.}

{k.akhir: idx berisi indeks A yang berisi nilai x. Jika x
tidak ditemukan, maka idx diisi dengan nilai -1}

var

    i : integer {pencatat indeks array}

begin

    i := 1;

    while (i < n) and (A[i] <> x) do

        i := i + 1;

    {endwhile}

    {i = n or A[i] = x}

    if A[i] = x then      {x ditemukan}

        Ketemu ← true

    else

        ketemu ← false   {x tidak ada di dalam array A}

    {endif}

end;

```

### 2.1.2.2. Menanya

1. Pada kasus di atas, algoritma pencarian linear akan bermanfaat jika nomor telepon ada di halaman pertama buku telepon, tetapi bagaimana jika ada di halaman paling akhir? Bagaimana juga bila buku telepon anda jumlah halamannya banyak sekali, misal 250 halaman?
2. Bagaimana menuliskan fungsi pada prosedur algoritma liniersearch1?

### **2.1.2.3. Mencoba**

1. Dalam kasus di atas, anda akan mencari kata demi kata dari halaman-halaman yang ada pada buku telepon. Dari halaman pertama sampai halaman selanjutnya hingga ditemukan nomor telepon yang dimaksud. Hal tersebut juga berlaku jika anda mencari kata pada sebuah buku yang tebalnya 1000 halaman yang tidak dilengkapi dengan indeks pencarian kata. Anda harus membuka dan membaca halam per halaman hingga ditemukannya kata tersebut.
2. Buatlah fungsi atau prosedur untuk melakukan pencarian linear. Tuliskan hasilnya pada kotak dibawah.

#### 2.1.2.4. Mengasosiasi

Secara umum, algoritma pencarian data berjalan lambat. Waktu pencarian sebanding dengan banyaknya jumlah data yang ada. Misalkan pada array berukuran  $n$  elemen. Maka, pada kasus dimana pencarian  $x$  pada array tersebut,  $x$  tidak terdapat di dalam array atau  $x$  ditemukan pada elemen yang terakhir, kita harus melakukan perbandingan dengan seluruh elemen array, yang berarti jumlah perbandingan yang terjadi sebanyak  $n$  kali. Kita katakan bahwa waktu pencarian dengan algoritma pencarian linier sebanding dengan  $n$ . Algoritma pencarian linier akan sangat berguna bila digunakan untuk mencari data pada sekumpulan data yang sedikit. Kelemahan dari algoritma pencarian linier adalah mencari data pada sekumpulan data yang banyak, maka membutuhkan waktu yang lama.

Pada algoritma `liniersearch1` di atas, perbandingan  $x$  dengan elemen array dilakukan pada kondisi pengulangan. Apabila elemen array yang ke- $i$  tidak sama dengan  $x$  dan  $i$  belum sama dengan  $n$ , aktivitas perbandingan diteruskan berikutnya ( $i \leftarrow i + 1$ ).

Perbandingan dihentikan apabila  $A[i] = x$  atau indeks  $i$  sudah mencapai akhir array ( $i = n$ ). Perhatikan juga bahwa jika  $i$  sudah mencapai akhir array. Elemen terakhir ini belum dibandingkan dengan dengan  $x$ . Perbandingan elemen terakhir dilakukan bersama-sama dengan menyimpulkan hasil pencarian. Hasil pencarian disimpulkan di luar kalang `while-do` dengan `if (A[i] = x) then ...`

Pernyataan *if-then* ini juga sekaligus memeriksa apakah elemen terakhir,  $A[n]$ , sama dengan  $x$ . Jadi, pada algoritma `liniersearch1` di atas, elemen terakhir diperiksa secara khusus.

Jika pada array tidak terurut jumlah perbandingan elemen array maksimum  $n$  kali, maka pada array terurut (dengan asumsi distribusi elemen-elemen array adalah seragam atau *uniform*) hanya dibutuhkan rata-rata  $n/2$  kali perbandingan. Hal ini dikarenakan pada array terurut kita dapat segera menyimpulkan bahwa array  $x$  tidak terdapat di dalam array bila ditemukan elemen array yang lebih besar dari  $x$  (pada larray yang terurut menaik). Jadi dengan begitu, array yang elemen-elemennya sudah terurut dapat meningkatkan kinerja algoritma pencarian linier.

menjadi elemen yang ditambahkan ke dalam array. Sebaliknya, jika  $idx < n+1$  (yang berarti  $x$  ditemukan sebelum sentinel), maka hal itu berarti bahwa  $x$  sudah ada di dalam array  $A$  semula.

#### 2.1.3. Rangkuman

Jadi pada dasarnya algoritma pencarian merupakan langkah-langkah dalam suatu proses pencarian data tertentu dalam sekumpulan data yang mempunyai tipe yang sama. Algoritma pencarian dengan data yang dicari lebih dari satu, maka pencarian selesai bila salah satu data sudah ditemukan.

Algoritma pencarian linier adalah proses pencarian dengan membandingkan atau mencari data satu persatu secara linier. Algoritma pencarian linier tidak bagus untuk volume data yang besar.

Array yang sudah terurut dapat meningkatkan kinerja algoritma pencarian linier. Algoritma pencarian linier dengan sentinel adalah melakukan penambahan data pada elemen array. Data dapat simpulkan terdapat pada array jika  $idx < n+1$ .

#### 2.1.4. Tugas

1. Carilah algoritma pencarian di kehidupan sehari-hari, kemudian tulis algoritmanya! Dari algoritma yang diperoleh, tunjukkan algoritma mana yang merupakan algoritma pencarian linier!
2. Prosedur algoritma pencarian dapat dipanggil dari program utama atau dari prosedur lain. Misalkan kita asumsikan prosedur `liniersearch1` dipanggil dari program utama yang bertujuan untuk memeriksa keberadaan  $x$  di dalam array. Jika  $x$  terdapat di dalam array maka ditampilkan pesan “ditemukan！”, sebaliknya jika  $x$  tidak terdapat di dalam array maka ditampilkan pesan “tidak ditemukan！”.

#### Lembar Kreativitas Siswa.

Buatlah program sederhana untuk mencari data string yang terimpan dalam array of string.

## 2.2. Kegiatan Belajar 2. Pemanfaatan Pencarian Data dalam Aplikasi

**Alokasi Waktu : 2x45 Menit**

### 2.2.1. Tujuan Pembelajaran

Peserta didik mampu menggunakan algoritma pencarian linier dalam aplikasi.

### 2.2.2. Aktivitas Belajar Siswa

#### 2.2.2.1. Mengamati

Permasalahan 1.

Seorang pedagang buah menjual beraneka ragam jenis buah. Pedagang tersebut membeli buah yang dijualnya itu dari 5 pasar yang ada di kotanya. Jarak lokasi penjualannya dengan pasar tersebut masing-masing sebagai berikut:

Pasar	Jarak (km)	Ketersediaan buah pada suatu hari
		(belum diketahui oleh pedagang)
A	11	Jeruk, mangga, semangka, apel, dan sirsat
B	15	Mangga, apel, dan jambu
C	7,5	Jambu, mangga, dan semangka
D	20	Jambu, mangga, dan apel
E	5	Apel, jambu, dan alpukat

Pada suatu hari pedagang ingin membeli buah jeruk, mangga, dan apel untuk dagangannya pada satu lokasi pasar untuk mempermudah proses pengangkutan buah. Bantulah pedagang tersebut membeli dan menemukan barang dagangannya tersebut.

Dari permasalahan tersebut, kita dapat membantu pedagang dengan menggunakan algoritma pencarian linier. Berikut langkah-langkahnya:

- Pertama kita harus menentukan urutan pasar yang akan dikunjungi tersebut. Dalam masalah tersebut dapat membuat urutan berdasarkan jarak terdekat terlebih dahulu. Urutannya sebagai berikut:

Pasar	E	C	A	B	D
Jarak	5	7,5	11	15	20
ketersediaan	Apel, jambu, dan alpukat	Jambu, mangga, dan semangka	Jeruk, mangga, semangka, apel, dan	Mangga, apel, dan jambu	Jambu, mangga, dan apel

			sirsat		
--	--	--	--------	--	--

2. Langkah berikutnya kita harus mengunjungi satu persatu secara beruntun masing-masing pasar sesuai dengan urutan yang kita buat tadi, kemudian kita cek ketersediaan buah pada pasar tersebut dan berhenti jika kita mendapat pasar yang menyediakan semua buah yang kita perlukan.
3. Langkah pertama ke pasar E, cek ketersediaan. Tidak ada mangga dan jeruk.
4. Selanjutnya ke pasar C, cek ketersediaan. Tidak ada jeruk dan apel.
5. Selanjutnya ke pasar A, cek ketersediaan. Ada
6. Berhenti dan membeli buah tersebut.

Dari langkah-langkah yang telah kita lakukan dengan menggunakan algoritma pencarian beruntun tersebut, kita mengunjungi 3 pasar hingga didapatkan buah yang dicari. Dengan kata lain, kita melakukan perbandingan 3 kali.

#### 2.2.2.2. Menanya

Bagaimana jika pasar buahnya ada 10 pasar?

#### 2.2.2.3. Mencoba

Misalkan klasifikasi pasarnya sebagai berikut:

Pasar	Jarak (km)	Ketersediaan buah pada suatu hari	
		(belum diketahui oleh pedagang)	
A	11	Manggis	dan Apel
B	15	Mangga, apel,	dan jambu
C	7,5	Jambu, mangga,	dan semangka
D	20	Jambu, mangga,	dan apel
E	5	Apel, jambu,	dan alpukat
F	13	Jeruk, Manggis,	dan Apel
G	9	Apel, jeruk,	dan manggis
H	6	Mangga, Apel,	dan salak
I	10	Salak	dan Manggis
J	19	Jeruk, mangga,	semangka, apel, dan sirsat

1. Langkah pertama buatlah urutan pasar yang akan dikunjungi sesuai kriteriamu.

Pasar	...	...	...	...	...	...	...	...	...	...
Jarak	...	...	...	...	...	...	...	...	...	...
ketersediaan	...	...	...	...	...	...	...	...	...	...

2. Langkah selanjutnya ke pasar ... pertama, cek ketersediaan. Jika tidak ada lanjut ke pasar berikutnya.
3. Langkah selanjutnya ke pasar ..., cek ketersediaan. Jika tidak ada lanjut ke pasar berikutnya.
4. ...
- :
- n. langkah ke n hingga ditemukan buah yang dicari. Pencarian paling banyak hingga 10 kali pencarian.

Buatlah dalam program dengan bahasa Pascal dan tuliskan hasilnya pada kotak dibawah.

#### **2.2.2.4. Mengasosiasi**

Pada permasalah menanya di atas dilakukan pencarian dengan algoritma pencarian linier yang memiliki kemungkinan terburuk melakukan perbandingan 10 kali. Tetapi untuk permasalahan 1, memiliki kemungkinan terburuk melakukan perbandingan 5 kali. Hal ini dikarenakan pencarian pada pasar yang berbeda jumlahnya. Dari hal tersebut dapat dilihat, bahwa melakukan pencarian dengan penggunaan algoritma pencarian beruntun, jika semakin besar kumpulan data yang dicari, maka makin banyak juga proses atau perbandingannya.

Sebenarnya, menggunakan algoritma pencarian linier pada aplikasi langkah awal yang harus dilakukan adalah membuat urutan data yang akan dibandingkan. Kemudian baru membandingkan data satu per satu secara beruntun dari data pertama hingga data terakhir.

#### **2.2.3. Rangkuman**

Penggunaan algoritma pencarian linier pada aplikasi sangat bermanfaat jika data yang dicari ada pada sekumpulan sedikit data. Jika pada data besar, pencarian ini tidak praktis dan tidak efisien

#### **2.2.4. Tugas**

Carilah manfaat dari penggunaan algoritma pencarian beruntun dalam kehidupan sehari-hari

## 2.3. Kegiatan Belajar 3. Pengurutan Data Dengan Algoritma *Bubble Sort*

Alokasi Waktu : 2x45 Menit

### 2.3.1. Tujuan Pembelajaran

1. Peserta didik mampu memahami algoritma pengurutan.
2. Peserta didik mampu membuat algoritma pengurutan gelembung.

### 2.3.2. Aktivitas Belajar Siswa

#### 2.3.2.1. Mengamati

Pengurutan (*sorting*) adalah proses mengatur sekumpulan objek menurut urutan atau susunan tertentu. Masalah pengurutan dapat ditulis sebagai berikut:

Diberikan array A dengan n elemen yang sudah terdefinisi elemen-elemennya.

Urutan array sehingga tersusun secara menaik (*ascending*):

$$A[1] \leq A[2] \leq A[3] \leq \dots \leq A[n]$$

Atau secara menurun (*descending*):

$$A[1] \geq A[2] \geq A[3] \geq \dots \geq A[n]$$

Dalam kehidupan sehari-hari, kata di dalam kamus bahasa/istilah dan entry di dalam ensiklopedi selalu terurut secara alfabetik. Kita juga sering melakukan pengurutan seperti mencatat nomor hp teman berdasarkan nama, menyusun tumpukan koran berdasarkan tanggal, mengantre berdasarkan waktu kedatangan, dan sebagainya. Data terurut memiliki beberapa keuntungan. Selain mempercepat waktu pencarian, dari data yang terurut kita dapat langsung memperoleh nilai maksimum dan nilai minimum. Untuk data numerik yang terurut menurun, nilai maksimum adalah elemen yang pertama array, dan nilai minimum adalah elemen terakhir. Dan sebaliknya untuk data numerik yang terurut menaik. Pengurutan dikatakan stabil jika ada dua atau lebih data yang sama (atau identik) tetap pada urutan yang sama setelah pengurutan. Misalnya di dalam sekelompok data integer berikut terdapat nilai 9 (diberi tanda petik ‘, “ ,” untuk mengidentifikasi urutannya).

38, 70, 9’, 10, 9”, 3, 8, 61, 9”

Jika metode pengurutan menghasilkan susunan data seperti berikut:

3, 8, 9’, 9”, 9”, 10, 38, 61, 70

Maka pengurutan dikatakan stabil dan metode pengurutannya disebut metode stabil. Tetapi jika suatu metode pengurutan menghasilkan susunan data terurut seperti di bawah ini:

3, 8, 9”, 9’, 9”, 10, 38, 61, 70

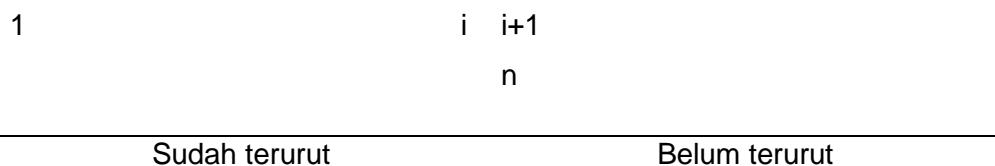
Maka pengurutan dan metodenya kita katakan tidak stabil.

Adanya kebutuhan terhadap proses pengurutan memunculkan bermacam-macam algoritma pengurutan. Banyak algoritma pengurutan yang telah ditemukan. Hal ini menunjukkan bahwa persoalan pengurutan adalah persoalan yang kaya dengan solusi algoritmik. Salah satu algoritma pengurutan adalah algoritma gelembung (*bubble sort*) dan algoritma seleksi (*selection sort*).

### Algoritma Bubble Sort

Algoritma *bubble sort* (pengurutan gelembung atau apung) diinspirasi oleh gelembung sabun yang berada di atas permukaan air. Karena berat jenis gelembung sabun lebih ringan daripada berat jenis air, maka gelembung sabun selalu terapung ke atas permukaan. Secara umum benda-benda yang ringan akan terapung ke atas permukaan.

Prinsip pengapungan di atas juga digunakan pada pengurutan apung. Apabila kita menginginkan array terurut menaik, maka elemen array yang berharga paling kecil “digelembungkan” atau “diapungkan”, artinya diangkat ke “atas” (atau ke ujung kiri array) melalui proses pertukaran. Proses pengapungan ini dilakukan sebanyak  $n-1$  langkah (satu langkah juga disebut satu kali pass) dengan  $n$  adalah ukuran array. Pada akhir setiap langkah ke  $i$ , array  $A[1..n]$  akan terdiri atas dua bagian yaitu bagian yang terurut, yaitu  $A[1..i]$ , dan bagian yang belum terurut,  $A[i+1..n]$  (gambar 6. 3). Setelah langkah terakhir didapat array  $A[1..n]$  yang terurut menaik.



**Gambar 2.3** Bagian array yang terurut dan belum terurut pada *bubble sort*

Untuk mendapatkan array yang terurut menaik, algoritma pengurutan gelembung secara global adalah sebagai berikut:

Untuk setiap pass  $i = 1, 2, \dots, n-1$ , dilakukan:

Mulai dari elemen  $k = n, n-1, \dots, i+1$ , dilakukan:

1. Bandingkan  $A[k]$  dengan  $A[k-1]$ .
2. Pertukarkan  $A[k]$  dengan  $A[k-1]$  jika  $A[k] < A[k-1]$

Rincian setiap pass sebagai berikut:

Pass 1 : Mulai dari elemen ke- $k = n, n-1, \dots, 2$ , bandingkan  $A[k]$  dengan  $A[k-1]$ . Jika  $A[k] < A[k-1]$ , pertukarkan  $A[k]$  dengan  $A[k-1]$ . Pada akhir langkah 1, elemen  $A[1]$  berisi harga minimum pertama.

Pass 2 : Mulai dari elemen ke- $k = n, n-1, \dots, 3$ , bandingkan  $A[k]$  dengan  $A[k-1]$ . Jika  $A[k] < A[k-1]$ , pertukarkan  $A[k]$  dengan  $A[k-1]$ . Pada akhir langkah 2, array  $A[1..2]$  terurut, sedangkan  $A[3..n]$  belum terurut.

Pass 3 : Mulai dari elemen ke- $k = n, n-1, \dots, 4$ , bandingkan  $A[k]$  dengan  $A[k-1]$ . Jika  $A[k] < A[k-1]$ , pertukarkan  $A[k]$  dengan  $A[k-1]$ . Pada akhir langkah 3, array  $A[1..3]$  terurut, sedangkan  $A[4..n]$  belum terurut.

Pass  $n-1$  : Mulai dari elemen ke- $k = n$  bandingkan  $A[k]$  dengan  $A[k-1]$ . Jika  $A[k] < A[k-1]$ , pertukarkan  $A[k]$  dengan  $A[k-1]$ . Pada akhir langkah  $n-1$ , array  $A[n-1]$  berisi nilai minimum ke  $(n-1)$  dan array  $A[1..n-1]$  terurut menaik (elemen yang tersisa adalah  $A[n]$ , tidak perlu diurutkan karena hanya satu-satunya).

Perhatikan array  $n = 6$  buah elemen berikut yang belum terurut. Array ini akan diurutkan menaik:

77	37	58	5	35	10
1	2	3	4	5	6

### Pass 1:

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
k = 6	A[6] < A[5] ? (10 < 35 ?)	ya	77, 37, 58, 5, <u>10, 35</u>
k = 5	A[5] < A[4] ? (10 < 5 ?)	tidak	77, 37, 58, <u>5, 10, 35</u>
k = 4	A[4] < A[3] ? (5 < 58 ?)	Ya	77, 37, 5, <u>58, 10, 35</u>
k = 3	A[3] < A[2] ? (5 < 37 ?)	Ya	77, 5, <u>37, 58, 10, 35</u>
k = 2	A[2] < A[1] ? (5 < 77 ?)	Ya	5, <u>77, 37, 58, 10, 35</u>

### Hasil akhir pass 1:

5	77	37	58	10	35
1	2	3	4	5	6

### Pass 2:

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
k = 6	A[6] < A[5] ? (35 < 10 ?)	Tidak	5, 77, 37, 58, <u>10, 35</u>
k = 5	A[5] < A[4] ? (10 < 58 ?)	Ya	5, 77, 37, <u>10, 58, 35</u>
k = 4	A[4] < A[3] ? (10 < 37 ?)	Ya	5, 77, <u>10, 37, 58, 35</u>
k = 3	A[3] < A[2] ? (10 < 77 ?)	Ya	<u>5, 10, 77, 37, 58, 35</u>

### Hasil akhir pass 2:

5	10	77	37	58	35
1	2	3	4	5	6

**Pass 3:**

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
k = 6	A[6] < A[5] ? (35 < 58 ?)	ya	5, 10, 77, 37, <u>35, 58</u>
k = 5	A[5] < A[4] ? (35 < 37 ?)	Ya	5, 10, 77, <u>35, 37, 58</u>
k = 4	A[4] < A[3] ? (35 < 77 ?)	ya	5, 10, <u>35, 77, 37, 58</u>

Hasil akhir pass 3:

5	10	35	77	37	58
---	----	----	----	----	----

**Pass 4:**

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
k = 6	A[6] < A[5] ? (58 < 37 ?)	tidak	5, 10, 35, 77, <u>37, 58</u>
k = 5	A[5] < A[4] ? (37 < 77 ?)	Ya	5, 10, 35, <u>37, 77, 58</u>

Hasil akhir pass 4:

5	10	35	37	77	58
---	----	----	----	----	----

**Pass 5:**

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
k = 6	A[6] < A[5] ? (58 < 77 ?)	Ya	5, 10, 35, 37, <u>58, 77</u>

Hasil akhir pass 5:

5	10	35	37	58	77
---	----	----	----	----	----

Berikut adalah penyajian penulisan algoritma pengurutan gelembung:

```
Procedure bubblesort1 (input/output A : arrayint, input n
: integer)

{mengurutkan array A[1..n] sehingga terurut menaik dengan
```

```

metode pengurutan gelembung}

{k.awal: elemen array A sudah terdefinisi nilai-nilainya.}

{k.akhir: elemen array A terurut menaik sedemikian
sehingga A[1] ≤ A[2] ≤ ... ≤ A[n].}

```

Deklarasikan

```

i      : integer {pencacah untuk jumlah langkah}

k      : integer {pencacah untuk penggelembung pada
setiap langkah}

temp  : integer {peubah bantu untuk pertukaran}

```

Algoritma:

```

for i ← 1 to n - 1 do

    for k ← n downto i + 1 do

        if A[k] < A[k - 1] then

            {pertukarkan A[k] dengan A[k - 1]}

            temp ← A[k]

            A[k] ← A[k - 1]

            A[k - 1] ← temp

        endif

    endfor

endfor

```

Kita dapat menuliskan proses pertukaran ( $\text{temp} \leftarrow A[k]$ ,  $A[k] \leftarrow A[k-1]$ ,  $A[k-1] \leftarrow \text{temp}$ ) sebagai sebuah prosedur tukar sebagai berikut:

```

Procedure tukar (input/output a : integer, input/output a
: integer)

{mempertukarkan nilai a dan b}

{k.awal: a dan b sudah terdefinisi nilai-nilainya.}

{k.akhir: b berisi nilai a sebelum pertukaran, dan b
berisi nilai a sebelum pertukaran.}

```

Deklarasikan

```
temp : integer {peubah bantu untuk pertukaran}
```

Algoritma:

```

temp ← a

a ← b

b ← temp

```

Dengan memanfaatkan prosedur tukar, algoritma pengurutan gelembung dapat disederhanakan penulisannya sebagai berikut:

```

Procedure bubblesort1 (input/output A : arrayint, input n
: integer)

{mengurutkan array A[1..n] sehingga terurut menaik dengan
metode pengurutan gelembung}

{k.awal: elemen array A sudah terdefinisi nilai-nilainya.}

{k.akhir: elemen array A terurut menaik sedemikian
sehingga A[1] ≤ A[2] ≤ ... ≤ A[n].}

```

Deklarasikan

```

i      : integer {pencacah untuk jumlah langkah}

k      : integer {pencacah untuk penggelembung pada

```

```

setiap langkah}

Procedure tukar (input/output a : integer, input/output b : integer)
{mempertukarkan nilai a dan b}

Algoritma:

for i ← 1 to n - 1 do

    for k ← n downto i + 1 do

        if A[k] < A[k - 1] then

            {pertukarkan A[k] dengan A[k - 1]}

            Tukar (A[k], A[k - 1])

        Endif

    endfor

endfor

```

Penulisan freepascal algoritma pengurutan gelembung di atas:

```

(*deklarasi*)

const Nmaks = 1000; {jumlah maksimum elemen array}

type arrayint = array [1..Nmaks] of integer;

Procedure bubblesort1 (var A : arrayint; input n : integer;)

{mengurutkan array A[1..n] sehingga terurut menaik dengan
metode pengurutan gelembung}

```

```
{k.awal: elemen array A sudah terdefinisi nilai-nilainya.}

{k.akhir: elemen array A terurut menaik sedemikian
sehingga A[1] ≤ A[2] ≤ ... ≤ A[n].}

var

    i      : integer; {pencacah untuk jumlah langkah}

    k      : integer;      {pencacah untuk penggelembung
pada setiap langkah}

    temp  : integer; {peubah bantu untuk pertukaran}

begin

    for i := 1 to n - 1 do

        for k := n downto i + 1 do

            if A[k] < A[k - 1] then

                {pertukarkan A[k] dengan A[k - 1]}

                temp      := A[k];

                A[k]      := A[k-1];

                A[k-1]    := temp;

            end; {if}

        {endfor}

    {endfor}

end;
```

### 2.3.2.2. Menanya

1. Bagaimana jika array memiliki elemen yang banyak, misal 10 elemen seperti array dibawah ini?

54	98	23	12	86	9	24	57	6	19
1	2	3	4	5	6	7	8	9	10

2. Bagaimana jika pengurutannya menurun? Bagaimana langkah-langkahnya?
3. Bagaimana penulisan prosedure algoritma jika harus memperoleh pengurutan menurun?

### 2.3.2.3. Mencoba

#### Percobaan 1

Array dengan  $n = 10$  buah elemen berikut yang belum terurut. Array ini akan diurutkan menaik:

54	98	23	12	86	9	24	57	6	19
1	2	3	4	5	6	7	8	9	10

← arah perbandingan

#### Pass 1:

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10$	$A[10] < A[9] ? (19 < 6 ?)$	tidak	54, 98, 23, 12, 86, 9, 24, 57, <u>6, 19</u>
$k = 9$	...	...	...
$k = 8$	...	...	...
$k = 7$	...	...	...
$k = 6$	...	...	...
$k = 5$	...	...	...
$k = 4$	...	...	...
$k = 3$	$A[3] < A[2] ? (\dots < \dots ?)$	...	...

$k = 2 \quad A[2] < A[1] ? (\dots < \dots ?)$  ... 6, ..., ..., ..., ..., 19

Hasil akhir pass 1:

6	...	...	...	...	...	...	...	57	19
1	2	3	4	5	6	7	8	9	10

**Pass 2:**

$k$	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10 \quad A[10] < A[9] ? (19 < 57 ?)$	...	Ya	6, ..., ..., ..., ..., 19, 57
$k = 9$	...	...	...
$k = 8$	...	...	...
$k = 7$	...	...	...
$k = 6$	...	...	...
$k = 5$	...	...	...
$k = 4$	...	...	...
$k = 3 \quad A[3] < A[2] ? (\dots < \dots ?)$	...	...	6, ..., ..., ..., ..., 57

Hasil akhir pass 2:

6	9	...	...	...	...	...	...	...	57
1	2	3	4	5	6	7	8	9	10

**Pass 3:**

$k$	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10 \quad A[10] < A[9] ? ( \dots < \dots ?)$	...	...	6, 9, ..., ..., ..., ..., ...
$k = 9$	...	...	...
$k = 8$	...	...	...
$k = 7$	...	...	...

$k = 6$	...	...	...
$k = 5$	...	...	...
$k = 4$	$A[4] < A[3] ? (\dots < \dots ?)$	...	...

Hasil akhir pass 3:

6	9	...	...	...	...	...	...	...	...
1	2	3	4	5	6	7	8	9	10

**Pass 4:**

$k$	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10$	$A[10] < A[9] ? (\dots < \dots ?)$	...	6, 9, ..., ....., ..., ...
$k = 9$	....	...	...
$k = 8$	...	...	...
$k = 7$	....	...	...
$k = 6$	...	...	...
$k = 5$	$A[5] < A[4] ? (\dots < \dots ?)$	...	...

Hasil akhir pass 4:

6	9	...	...	...	...	...	...	...	...
1	2	3	4	5	6	7	8	9	10

**Pass 5:**

$k$	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10$	$A[10] < A[9] ? (\dots < \dots ?)$	...	6, 9, ..., ....., ..., ...
$k = 9$	....	...	...

$k = 8$	...	...	...
$k = 7$	....	...	...
$k = 6$	...	...	...

Hasil akhir pass 5:

6	9	...	...	...	...	...	...	...	...	...
1	2	3	4	5	6	7	8	9	10	

**Pass 6:**

$k$	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10$	$A[10] < A[9] ? ( \dots < \dots ? )$	...	6, 9, ..., ....., ..., ....
$k = 9$	....	...	...
$k = 8$	...	...	...
$k = 7$	....	...	...

Hasil akhir pass 6:

6	9	...	...	...	...	...	...	...	...	...
1	2	3	4	5	6	7	8	9	10	

**Pass 7:**

$k$	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10$	$A[10] < A[9] ? ( \dots < \dots ? )$	...	6, 9, ..., ....., ..., ....
$k = 9$	....	...	...
$k = 8$	...	...	...

Hasil akhir pass 7:

6	9	...	...	...	...	...	...	...	...
1	2	3	4	5	6	7	8	9	10

**Pass 8:**

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10$	$A[10] < A[9] ? ( \dots < \dots ? )$	...	6, 9, ..., ...., ..., ....
$k = 9$	....	...	...

Hasil akhir pass 8:

6	9	...	...	...	...	...	...	...	...
1	2	3	4	5	6	7	8	9	10

**Pass 9:**

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10$	$A[10] < A[9] ? ( \dots < \dots ? )$	...	6, 9, ..., ...., ..., ....

Hasil akhir pass 9:

6	9	...	...	...	...	...	...	...	...
1	2	3	4	5	6	7	8	9	10

Ada 9 pass (langkah),  $\text{pass} = n - 1$ ,  $10 - 1 = 9$ .**Percobaan 2**

Untuk mendapatkan array yang terurut menurun, algoritma untuk menguratkannya adalah sebagai berikut:

Untuk setiap pass  $i = 1, 2, \dots, n-1$  lakukan:

Mulai dari elemen  $k = \dots, \dots, \dots$ , lakukan:

1. Bandingkan  $A[k]$  dengan  $A[k-1]$ .
2. Pertukarkan  $A[k]$  dengan  $A[k-1]$  jika  $A[k] > A[k-1]$ .

Rincian setiap pass sebagai berikut:

Pass 1 : Mulai dari elemen ke- $k = n, n-1, \dots, 2$ , bandingkan  $A[k]$  dengan  $A[k-1]$ . Jika  $A[k] (\dots) A[k-1]$ , pertukarkan  $A[k]$  dengan  $A[k-1]$ . Pada akhir langkah 1, elemen  $A[1]$  berisi harga .....

Pass 2 : Mulai dari elemen ke- $k = n, n-1, \dots, 3$ , bandingkan  $A[k]$  dengan  $A[k-1]$ . Jika  $A[k] (\dots) A[k-1]$ , pertukarkan  $A[k]$  dengan  $A[k-1]$ . Pada akhir langkah 2, array  $A[1..2]$  terurut, sedangkan  $A[3..n]$  belum terurut.

Pass 3 : ...

Pass 4 : ...

Pass  $n-1$  : Mulai dari elemen ke- $k = n$  bandingkan  $A[k]$  dengan  $A[k-1]$ . Jika  $A[k] (\dots) A[k-1]$ , pertukarkan  $A[k]$  dengan  $A[k-1]$ . Pada akhir langkah  $n-1$ , array  $A[n-1]$  berisi nilai ... ke  $(n-1)$  dan array  $A[1..n-1]$  terurut ....

\*Isilah bagian yang kurang (...).

### Percobaan 3

Buatlah program.

```
Procedure bubblesort1 (input/output A : arrayint, input n
: integer)

{mengurutkan array A[1..n] sehingga terurut menurun
dengan metode pengurutan gelembung}

{k.awal    : ....}

{k.akhir   : ....}
```

Deklarasikan

```
i      : integer {pencacah untuk jumlah langkah}

k      : integer {pencacah untuk penggelembung pada
setiap langkah}
```

```
Procedure tukar (input/output a : integer, input/output b
: integer)

{mempertukarkan nilai a dan b}
```

```
Algoritma:  
  
    for ... to ... do  
  
        for ... downto ... do  
  
            if ... then  
  
                {pertukarkan A[k] dengan A[k - 1]}  
  
                Tukar (A[k], A[k - 1])  
  
            endif  
  
        endfor  
  
    endfor
```

#### 2.3.2.4. Mengasosiasi

Pada algoritma gelembung atau apung (*bubble sort*) merupakan algoritma yang tidak efisien. Hal ini disebabkan jika elemen yang dimiliki array banyak, maka langkah-langkah (pass) dan operasi pertukaran yang dilakukan setiap langkahnya akan semakin banyak juga sehingga membutuhkan waktu yang lama. Namun, kelebihan dari algoritma gelembung adalah mudah dipahami.

Untuk melakukan pengurutan dengan algoritma gelembung, yang membedakan pengurutan menaik dan pengurutan menurun adalah hanya dari perbandingannya antara kurang dari ( $<$ ) dan lebih dari ( $>$ ). Elemen  $A[1]$  pada pengurutan menaik berisi elemen dengan harga minimum, dan untuk elemen  $A[1]$  pada pengurutan menurun berisi elemen dengan harga maksimum.

Untuk mendapatkan atau memperoleh array terurut menurun, kita melakukan proses sebaliknya, yaitu “melempar” elemen berharga maksimum ke “atas” (atau ke ujung “kiri” array).

#### 2.3.3. Rangkuman

Algoritma pengurutan (*sorting*) adalah algoritma atau proses mengatur sekumpulan objek menurut urutan atau susunan tertentu. Inti dari algoritma pengurutan gelembung atau apung (*bubble sort*) mengapungkan harga suatu array, bila pengurutan menaik, maka harga yang kecil yang “diapungkan” atau “digelembungkan”, dan jika pengurutan menurun, harga yang besar yang “diapungkan” atau “digelembungkan”. Pada pengurutan menaik, elemen pertama berisi harga paling minimum dan pada pengurutan menurun, elemen pertama berisi harga paling maksimum.

#### 2.3.4. Tugas

Carilah dalam kegiatan sehari-hari tentang kegiatan mengurutkan yang menggunakan algoritma pengurutan dengan metode gelembung (*bubble sort*).

### 2.3.5. Uji kompetensi

1. Apa yang dimaksud dengan algoritma pengurutan?
2. Apa yang dimaksud dengan algoritma pengurutan gelembung?
3. Apa yang menjadi ciri khas dari algoritma pengurutan gelembung?
4. Buatlah penyelesaian untuk mendapatkan array menaik dengan menggunakan algoritma gelembung dari array berikut:

56	76	2	1	9
1	2	3	4	5

15	71	98	5	42	6	9	20	63
1	2	3	4	5	6	7	8	9

5. Pada soal d. buatlah penyelesaian untuk mendapatkan array menurun dengan menggunakan algoritma gelembung!
6. Tulislah kembali algoritma pengurutan gelembung sedemikian sehingga elemen-elemen terurut “tumbuh” dari “kanan” ke “kiri” (atau dari “bawah” ke “atas”)!

## 2.4. Kegiatan Belajar 4. Pengurutan Data Dengan Algoritma *Selection Sort*

Alokasi Waktu : 2x45 Menit

### 2.4.1. Tujuan Pembelajaran

1. Peserta didik mampu memahami algoritma pengurutan seleksi.
2. Peserta didik mampu memahami algoritma seleksi-maksimum.
3. Peserta didik mampu memahami algoritma seleksi-minimum.

### 2.4.2. Aktivitas Belajar Siswa

#### 2.4.2.1. Mengamati

Algoritma pengurutan ini disebut pengurutan seleksi (*selection sort*) karena gagasan dasarnya adalah memilih elemen maksimum atau minimum dari array, lalu menempatkan elemen maksimum atau minimum itu pada awal atau akhir array (elemen terujung) (lihat gambar 6.4). selanjutnya elemen terujung disebut “diisolasi” dan tidak disertakan pada proses selanjutnya. Proses yang sama diulang untuk elemen array yang tersisa, yaitu memilih elemen maksimum atau minimum berikutnya dan mempertukarkannya dengan elemen terujung array tersisa. Semagaimna dengan algoritma pengurutan gelembung, proses memilih nilai maksimum atau minimum dilakukan pada setiap pass (langkah). Jika array berukuran  $n$ , maka jumlah pass adalah  $n - 1$ .

Sebelum:

1

n

---

Belum terurut

Sesudah:

1

n

---

Belum terurut

---

terurut

Gambar 6.4 bagian array yang terurut dan belum terurut pada algoritma pengurutan seleksi

Ada dua varian algoritma pengurutan seleksi ditinjau dari pemilihan elemen maksimum atau minimum, yaitu:

1. Algoritma pengurutan seleksi-maksimum, yaitu memilih elemen maksimum sebagai basis pengurutan.
2. Algoritma pengurutan seleksi-minimum, yaitu memilih elemen minimum sebagai basis pengurutan.

### Algoritma pengurutan seleksi-maksimum

Untuk mendapatkan array yang terurut menaik, algoritma pengurutan seleksi-maksimum secara garis besar ditulis sebagai berikut:

1. Jumlah pass =  $n - 1$
2. Untuk setiap pass = 1, 2, ...,  $n-1$ , lakukan:
  - a. Cari elemen terbesar (maks) mulai dari elemen ke-1 sampai elemen ke- $n$ .
  - b. Pertukarkan maks dengan elemen ke- $n$ .
  - c. Kurangi  $n$  satu (karena emelen ke- $n$  sudah terurut)

Rincian aksi setiap pass adalah sebagai berikut:

Pass 1 : cari elemen maksimum di dalam  $A[1..n]$ .

Pertukarkan elemen maksimum dengan elemen  $L[n]$ .

Ukuran array yang belum terurut =  $n - 1$ .

Pass 2 : cari elemen maksimum di dalam  $A[1..n - 1]$

Pertukarkan elemen maksimum dengan elemen  $L[n - 1]$

Ukuran array yang belum terurut =  $n - 2$

Pass 3 : cari elemen maksimum di dalam  $A[1..n - 2]$ .

Pertukarkan elemen maksimum dengan elemen  $L[n - 2]$ .

Ukuran array yang belum terurut =  $n - 3$ .

.

.

.

Pass  $n-1$  : cari elemen maksimum di dalam  $A[1..2]$

Pertukarkan elemen maksimum dengan elemen  $L[2]$

Ukuran array yang belum terurut = 2

Setelah pas  $n - 1$ , elemen yang tersisa adalah  $A[1]$ , tidak perlu diurutkan lagi karena hanya satu-satunya. Perhatikan array  $n = 6$  buah elemen berikut yang belum terurut. Array ini akan diurutkan menaik dengan metode pengurutan seleksi-maksimum:

77	37	58	5	35	10
1	2	3	4	5	6

#### Pass 1:

Carilah elemen maksimum di dalam array  $A[1..6] \rightarrow$  hasilnya: maks =  $A[2]$

= 77, pertukarkan maks dengan  $A[n]$  dalam hal ini  $A[6]$ , diperoleh:

Hasil akhir pass 1:

10	37	58	5	35	<b>77</b>
1	2	3	4	5	6

**Pass 2:**

(berdasarkan susunan array hasil pass 1)

Carilah elemen maksimum di dalam array  $A[1..5] \rightarrow$  hasilnya:  $maks = A[3] = 58$ , pertukarkan maks dengan  $A[5]$ , diperoleh:

Hasil akhir pass 2:

10	37	35	5	<b>58</b>	77
1	2	3	4	5	6

**Pass 3:**

(berdasarkan susunan array hasil pass 2)

Carilah elemen maksimum di dalam array  $A[1..4] \rightarrow$  hasilnya:  $maks = A[2] = 37$ , pertukarkan maks dengan  $A[4]$ , diperoleh:

Hasil akhir pass 3:

10	5	35	<b>37</b>	58	77
1	2	3	4	5	6

**Pass 4:**

(berdasarkan susunan array hasil pass 3)

Carilah elemen maksimum di dalam array  $A[1..3] \rightarrow$  hasilnya:  $maks = A[3] = 35$ , pertukarkan maks dengan  $A[3]$  (atau tetap, karena sudah sesuai lokasi elemennya), diperoleh:

Hasil akhir pass 4:

10	5	<b>35</b>	37	58	77
1	2	3	4	5	6

**Pass 5:**

(berdasarkan susunan array hasil pass 4)

Carilah elemen maksimum di dalam array  $A[1..2] \rightarrow$  hasilnya:  $maks = A[1] = 10$ , pertukarkan maks dengan  $A[2]$ , diperoleh:

Hasil akhir pass 5:

5	10	35	37	58	77
1	2	3	4	5	6

Tersisa satu elemen (yaitu 5), maka pengurutan selesai. Array A sudah terurut menaik.

Berikut adalah penyajian penulisan algoritma pengurutan seleksi-maksimum menaik

```
Procedure SelectionSort1 (input/output A : arrayint,
input n : integer)

{mengurutkan elemen-elemen array A[1..n] sehingga
terurut menaik dengan metode pengurutan seleksi-maksimum}

{k.awal: elemen array A sudah terdefinisi harganya.}

{k.akhir: elemen array A terurut menaik sedemikian
sehingga A[1] ≤ A[2] ≤ ... ≤ A[n].}
```

Deklarasikan

```
i      : integer {pencacah pass}

j      : integer {pencacah untuk mencari nilai
maksimum}

imaks : integer {indeks yang berisi nilai maksimum
sementara}

maks  : integer {elemen maksimum}

temp  : integer {peubah bantu untuk pertukaran}
```

Algoritma:

```
for i ← n downto 2 do {jumlah pass sebanyak n - 1}

          {cari elemen maksimum pada elemen A[1..i]}

          imaks ← 1 {elemen pertama diasumsikan sebagai
          elemen maksimum sementara}
```

```

maks ← A[1]           {elemen maksimum}

for j ← 2 to i do

    if A[j] > maks then

        imaks ← j

        maks ← A[j]

    endif

endfor

{pertukarkan maks dengan A[i]}

temp ← A[i]

A[i] ← maks

A[imaks] ← temp

endfor

```

Perhatikan bahwa pada algoritma selectionsor di atas kita dapat menghilangkan penggunaan peubah maks, karena yang kita perlukan sebenarnya adalah indeks elemen array yang mengandung nilai maksimum tersebut. Jika indeks itu kita catat, maka elemen arraynya dapat diacu melalui indeks tersebut. Dengan demikian, algoritma pengurutan seleksi-maksimum dapat diubah menjadi lebih elegan sebagai berikut:

```

Procedure selectionsort1 (input/output A : arrayint,
input n : integer)

{mengurutkan elemen-elemen array A[1..n] sehingga
terurut menaik dengan metode pengurutan selesi-maksimum}

{k.awal: elemen array A sudah terdefinisi harganya.}

{k.akhir: elemen array A terurut menaik sedemikian
sehingga A[1] ≤ A[2] ≤ ... ≤ A[n].}

```

Deklarasikan

```
i      : integer {pencacah pass}  
j      : integer {pencacah untuk mencari nilai  
maksimum}  
  
imaks : integer {indeks yang berisi nilai maksimum  
sementara}  
  
maks  : integer {elemen maksimum}  
  
temp  : integer {peubah bantu untuk pertukaran}
```

Algoritma:

```
for i ← n downto 2 do {jumlah pass sebanyak n - 1}  
  
    {cari elemen maksimum pada elemen A[1..i]}  
  
    imaks ← 1 {elemen pertama diasumsikan sebagai  
    elemen maksimum sementara}  
  
    for j ← 2 to i do  
  
        if A[j] > A[imaks] then  
  
            imaks ← j  
  
        endif  
  
    endfor  
  
    {pertukarkan A[imaks] dengan A[i]}  
  
    temp ← A[i]  
  
    A[i] ← A[imaks]  
  
    A[imaks] ← temp
```

```
endfor
```

seperti halnya pada pengurutan gelembung, prosedur tukar dapat digunakan untuk mengganti runtunan aksi pertukasan pada `selectionsort1` tadi. Berikut algoritma pengurutan seleksi-maksimum dengan prosedure tukar:

```
Procedure selectionsort1 (input/output A : arrayint,
input n : integer)

{mengurutkan elemen-elemen array A[1..n] sehingga
terurut menaik dengan metode pengurutan selesi-maksimum}

{k.awal: elemen array A sudah terdefinisi harganya.}

{k.akhir: elemen array A terurut menaik sedemikian
sehingga A[1] ≤ A[2] ≤ ... ≤ A[n].}
```

Deklarasikan

```
i      : integer {pencacah pass}

j      : integer {pencacah untuk mencari nilai
maksimum}

imaks : integer {indeks yang berisi nilai maksimum
sementara}
```

```
Procedure tukar (input/output a : integer, input/output
b : integer)

{mempertukarkan nilai a dan b}
```

Algoritma:

```
for i ← n downto 2 do {jumlah pass sebanyak n - 1}
```

```
{cari elemen maksimum pada elemen A[1..n]}
```

```
imaks ← 1 {elemen pertama diasumsikan sebagai  
elemen maksimum sementara}

for j ← 2 to i do

    if A[j] > A[imaks] then

        imaks ← j

    endif

endfor

{pertukarkan A[imaks] dengan A[i]}

Tukar(A[imaks], A[i])

endfor
```

**Penulisan freepascal algoritma pengurutan seleksi-maksimum di atas.**

```
(*deklarasi*)

const Nmaks = 1000; {jumlah maksimum elemen array}

type arrayint = array [1..Nmaks] of integer;
```

```
procedure selectionsort1 (var A : arrayint; input n :
integer;)

{mengurutkan elemen-elemen array A[1..n] sehingga
terurut menaik dengan metode pengurutan selesa-maksimum}

{k.awal: elemen array A sudah terdefinisi harganya.}

{k.akhir: elemen array A terurut menaik sedemikian
sehingga A[1] ≤ A[2] ≤ ... ≤ A[n].}
```

```
var

    i      : integer; {pencacah pass}

    j      : integer;      {pencacah untuk mencari nilai
maksimum}

    imaks : integer; {indeks yang berisi nilai maksimum
sementara}

    temp  : integer; {peubah bantu untuk pertukaran}

begin

for i := n downto 2 do {jumlah pass sebanyak n - 1}

    begin

        {cari elemen maksimum pada elemen A[1..i]}

        imaks := 1;      {elemen pertama diasumsikan
sebagai elemen maksimum sementara}

        for j := 2 to i do

            if A[j] > A[imaks] then

                imaks := j;

            {endif}

        {endfor}

        {pertukarkan A[imaks] dengan A[i] }

        temp := A[imaks];

        A[imaks] := A[i];

        A[i] := temp;

    end; {for}
```

```
end
```

#### 2.4.2.2. Menanya

1. Bagaimana jika array memiliki elemen yang banyak, misal 10 elemen seperti array dibawah ini?

54	98	23	12	86	9	24	57	6	19
1	2	3	4	5	6	7	8	9	10

2. Bagaimana jika pengurutannya menurun? Bagaimana langkah-langkahnya? Dan urutkan menurun array berikut:

77	37	58	5	35	10
1	2	3	4	5	6

3. Bagaimana prosedur algoritma selectionsort1 jika harus memperoleh terurut menurun?

#### 2.4.2.3. Mencoba

##### Percobaan 1

Array dengan  $n = 10$  buah elemen berikut yang belum terurut. Array ini akan diurutkan menaik dengan algoritma pengurutan seleksi-maksimum:

54	98	23	12	86	9	24	57	6	19
1	2	3	4	5	6	7	8	9	10

##### Pass 1:

Carilah elemen maksimum di dalam array  $A[1..10] \rightarrow$  hasilnya:  $maks = A[...] = ...$ , pertukarkan maks dengan  $A[...]$ , diperoleh:

Hasil akhir pass 1:

...	...	...	...	...	...	...	...	...	98
1	2	3	4	5	6	7	8	9	10

##### Pass 2:

Carilah elemen maksimum di dalam array  $A[...] \rightarrow$  hasilnya:  $maks = A[...] = ...$ , pertukarkan maks dengan  $A[...]$ , diperoleh:

Hasil akhir pass 2:

...	...	...	...	...	...	...	...	...	98
1	2	3	4	5	6	7	8	9	10

**Pass 3:**

Carilah elemen maksimum di dalam array  $A[...]$  → hasilnya:  $maks = A[...] = ...$ , pertukarkan maks dengan  $A[...]$ , diperoleh:

Hasil akhir pass 3:

...	...	...	...	...	...	...	...	98	
1	2	3	4	5	6	7	8	9	10

**Pass 4:**

Carilah elemen maksimum di dalam array  $A[...]$  → hasilnya:  $maks = A[...] = ...$ , pertukarkan maks dengan  $A[...]$ , diperoleh:

Hasil akhir pass 4:

...	...	...	...	...	...	...	...	98	
1	2	3	4	5	6	7	8	9	10

**Pass 5:**

Carilah elemen maksimum di dalam array  $A[...]$  → hasilnya:  $maks = A[...] = ...$ , pertukarkan maks dengan  $A[...]$ , diperoleh:

Hasil akhir pass 5:

...	...	...	...	...	...	...	...	98	
1	2	3	4	5	6	7	8	9	10

**Pass 6:**

Carilah elemen maksimum di dalam array  $A[...]$  → hasilnya:  $maks = A[...] = ...$ , pertukarkan maks dengan  $A[...]$ , diperoleh:

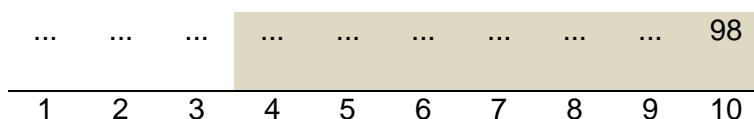
Hasil akhir pass 6:

...	...	...	...	...	...	...	...	98	
1	2	3	4	5	6	7	8	9	10

**Pass 7:**

Carilah elemen maksimum di dalam array  $A[...]$  → hasilnya:  $maks = A[...] = ...$ , pertukarkan maks dengan  $A[...]$ , diperoleh:

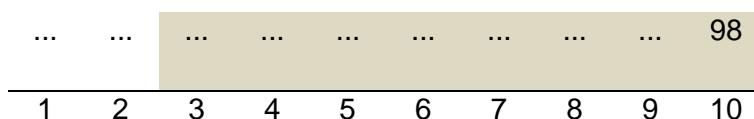
Hasil akhir pass 7:



**Pass 8:**

Carilah elemen maksimum di dalam array  $A[...]$  → hasilnya:  $maks = A[...] = ...$ , pertukarkan maks dengan  $A[...]$ , diperoleh:

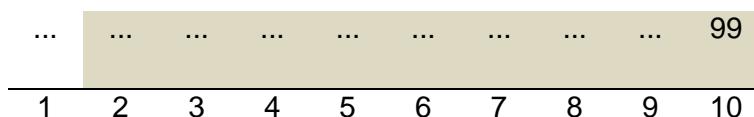
Hasil akhir pass 8:



**Pass 9:**

Carilah elemen maksimum di dalam array  $A[...]$  → hasilnya:  $maks = A[...] = ...$ , pertukarkan maks dengan  $A[...]$ , diperoleh:

Hasil akhir pass 9:



Ada 9 pass (langkah),  $\text{pass} = n - 1$ ,  $10 - 1 = 9$ .

## Percobaan 2

Untuk mendapatkan array yang terurut menurun, algoritma untuk mengurutkannya adalah sebagai berikut:

Jumlah pass =  $n - 1$

Untuk setiap pass = 1, 2, ...,  $n-1$ , lakukan:

- Cari elemen terbesar (maks) mulai dari elemen ke-1 sampai elemen ke- $n$ .
- Pertukarkan maks dengan elemen ke- ....

Rincian aksi setiap pass adalah sebagai berikut:

Pass 1 : cari elemen maksimum di dalam  $A[1..n]$ .

Pertukarkan elemen maksimum dengan elemen ....

Pass 2 : cari elemen maksimum di dalam ....

Pertukarkan elemen maksimum dengan elemen ....

Pass 3 : .....

.....

.

.

.

Pass n-1 : cari elemen maksimum di dalam ....

Pertukarkan elemen maksimum dengan elemen ....

Setelah pas n – 1, elemen yang tersisa adalah A[1], tidak perlu diurutkan lagi karena hanya satu-satunya.

### Percobaan 3

Prosedur algoritma selectionsrt1 jika harus memperoleh terurut menurun.

```
Procedure selectionsrt1 (input/output A : arrayint,
input n : integer)

{mengurutkan elemen-elemen array A[1..n] sehingga
terurut menurun dengan metode pengurutan selesi-
maksimum}

{k.awal    : ....}

{k.akhir   : ....}
```

Deklarasikan

```
i      : integer {pencacah pass}

j      : integer {pencacah untuk mencari nilai
maksimum}

imaks : integer {indeks yang berisi nilai maksimum
sementara}
```

```
Procedure tukar (input/output a : integer, input/output
```

```
b : integer)

{mempertukarkan nilai a dan b}

Algoritma:

for ... downto ... do {jumlah pass sebanyak n - 1}

    {cari elemen maksimum pada elemen A[1..n]}

        imaks ← 1 {elemen pertama diasumsikan sebagai
elemen maksimum sementara}

        for ... to ... do

            if ... then

                imaks ← j

            endif

        endfor

        {pertukarkan A[imaks] dengan A[i]}

        Tukar(..., ...)

    endfor
```

#### 2.4.2.4. Mengasosiasi

Algoritma pengurutan seleksi-maksimum merupakan algoritma yang cukup efisien, karena pada setiap pass-nya, operasi pertukaran hanya dilakukan sekali. Tetapi, kelemahan pada algoritma ini terletak pada banyaknya pass atau langkah. Untuk array yang berukuran besar, jumlah langkah atau pas juga besar, sehingga membutuhkan waktu yang cukup lama, tapi algoritma ini lebih baik jika dibandingkan dengan algoritma pencarian gelembung.

Untuk mendapatkan atau memperoleh array terurut menurun, kita melakukan proses sebaliknya, yaitu tanda kurang dari sama dengan diubah menjadai lebih dari sama dengan pada k.akhir.

Algoritma pengurutan seleksi minimum sebenarnya mirip dengan seleksi maksimum hanya berkebalikan saja, yaitu selalu dicari nilai minimum untuk setiap tahapnya.

#### 2.4.3. Rangkuman

Algoritma pengurutan seleksi adalah algoritma pengurutan yang berdasarkan pada pencarian elemen yang kemudian baru diurutkan. Ada dua varian algoritma pengurutan seleksi ditinjau dari pemilihan elemen maksimum atau minimum, yaitu:

1. Algoritma pengurutan seleksi-maksimum, yaitu memilih elemen maksimum sebagai basis pengurutan (pencarian elemen maksimum kemudian baru diurutkan).
2. Algoritma pengurutan seleksi-minimum, yaitu memilih elemen minimum sebagai basis pengurutan (pencarian elemen minimum kemudian baru diurutkan).

#### 2.4.4. Tugas

Tulislah kembali langkah-langkah melakukan pengurutan dengan menggunakan algoritma pengurutan seleksi-maksimum dan algoritma pengurutan seleksi minimum baik untuk urutan menaik maupun menurun!

#### 2.4.5. Uji Kompetensi

1. Apa yang dimaksud dengan algoritma pengurutan seleksi?
2. Apa yang dimaksud dengan algoritma pengurutan seleksi-maksimum?
3. Apa yang dimaksud dengan algoritma pengurutan seleksi-minimum?
4. Buatlah penyelesaian menggunakan algoritma pengurutan seleksi-maksimum dengan urutan menaik dan menurun dari array berikut:

56	76	2	1	9				
1	2	3	4	5				
15	71	98	5	42	6	9	20	63
1	2	3	4	5	6	7	8	9

5. Pada soal 4. buatlah penyelesaian pengurutan dengan menggunakan algoritma pengurutan seleksi-minimum dengan urutan menaik dan menurun!

6. Modifikasi algoritma pengurutan seleksi (baik yang minimum maupun maksimum) sedemikian sehingga jika elemen maksimum atau minimum yang ditemukan sudah pada posisi yang seharusnya, maka tidak perlu dilakukan pertukaran!

## 6.5. Kegiatan Belajar 5. Pemanfaatan Pengurutan dalam Aplikasi

**Alokasi Waktu : 2x45 Menit**

### 6.5.1. Tujuan Pembelajaran

1. Peserta didik mampu menggunakan algoritma pengurutan gelembung dalam aplikasi.
2. Peserta didik mampu menggunakan algoritma pengurutan selektion dalam aplikasi.

### 6.5.2. Aktivitas Belajar Siswa

#### 6.5.2.1. Mengamati

Permasalahan 1.

Sebuah perusahaan ingin membuat peringkat penjualan untuk melihat keuntungan yang didapat dari masing-masing unit penjualan setiap bulannya. Peringkat berdasarkan banyaknya barang yang dijual dalam setiap unitnya. Berikut adalah tabel penjualan dari masing-masing unit penjualan.

Unit penjualan	Jumlah penjualan (barang) per bulan
A	1.409
B	398
C	1.004
D	958
E	500

Dari permasalahan tersebut, perusahaan dapat mengurutkan peringkat dari tinggi ke rendah atau banyak ke sedikit (menurun) menggunakan algoritma pengurutan gelembung sebagai berikut:

1. Langkah 1.

UNIT	A	B	C	D	E
JUMLAH	1.409	398	1.004	958	500
	1	2	3	4	5

K	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
k = 5	A[5] > A[4] (E > D, tidak)	tidak	A, B, C, <u>D, E</u>
k = 4	A[4] > A[3] (D > C, tidak)	tidak	A, B, <u>C, D, E</u>
k = 3	A[3] > A[2] (C > B, ya )	Ya	A, <u>C, B, D, E</u>
k = 2	A[2] > A[1] (C > A tidak)	tidak	<u>A, C, B, D, E</u>

Hasil akhir langkah 1:

UNIT	A	C	B	D	E
JUMLAH	1.409	1.004	398	958	500
	1	2	3	4	5

## 2. Langkah 2.

K	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
k = 5	A[5] > A[4] (E > D, tidak)	tidak	A, C, B, <u>D, E</u>
k = 4	A[4] > A[3] (D > B, ya)	ya	A, C, <u>D, B, E</u>
k = 3	A[3] > A[2] (D > C, tidak )	tidak	A, <u>C, D, B, E</u>

Hasil akhir langkah 2:

UNIT	A	C	D	B	E
JUMLAH	1.409	1.004	958	398	500
	1	2	3	4	5

## 3. Langkah 3

K	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
k = 5	A[5] > A[4] (E > B, ya)	ya	A, C, D, <u>E, B</u>
k = 4	A[4] > A[3] (E > D, tidak)	tidak	A, C, <u>D, E, B</u>

Hasil akhir langkah 3:

UNIT	A	C	D	E	B
JUMLAH	1.409	1.004	958	500	398

---

 1      2      3      4      5

## 4. Langkah 4

K	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 5$	$A[5] > A[4]$ ( $E > B$ , tidak)	tidak	A, C, D, <u>E</u> , B

Hasil akhir langkah 4:

UNIT	A	C	D	E	B
JUMLAH	1.409	1.004	958	500	398
	1	2	3	4	5

Hasil dari pengurutan

Unit penjualan	Jumlah penjualan (barang) per bulan
A	1.409
C	1.004
D	958
E	500
B	398

Dari langkah-langkah yang telah kita lakukan dengan menggunakan algoritma pengurutan gelembung, dengan melakukan langkah 4 kalidan menghasilkan peringkat penjualan dari yang tertinggi ke terendah.

**2.5.2.2. Menanya**

Dari permasalahan di atas, bagaimana jika unit penjualan yang dimiliki perusahaan ada 10 unit?

**2.5.2.3. Mencoba**

Misalkan tabel unit dan jumlah penjualan adalah sebagai berikut:

Unit penjualan	Jumlah penjualan (barang) per bulan
A	1.409
B	398

C	1.004
D	958
E	500
F	647
G	71
H	907
I	1590
J	479

## 1. Langkah pertama.

UNIT	A	B	C	D	E	F	G	H	I	J
JUMLAH	1.409	398	1.004	958	500	647	71	907	1590	479
	1	2	3	4	5	6	7	8	9	10

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
k = 10	A[10] > A[9] (J > I tidak)	tidak	..., ..., ..., ..., <u>I, J</u>
k = 9	...	...	...
k = 8	...	...	...
k = 7	....	...	...
k = 6	...	...	...
k = 5	...	...	...
k = 4	...	...	...
k = 3	A[3] > A[2] ? (... > ... ?)	...	...

Hasil akhir langkah 2:

UNIT G ... ... ... ... ... ... ... ... J

JUMLAH	71	...	...	...	...	...	...	...	...	...	479
		1	2	3	4	5	6	7	8	9	10

## 2. Langkah 2

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10$	$A[10] > A[9]$ ( $J > I$ tidak)	tidak	$\dots, \dots, \dots, \dots, \underline{I}, J$
$k = 9$	...	...	...
$k = 8$	...	...	...
$k = 7$	...	...	...
$k = 6$	...	...	...
$k = 5$	...	...	...
$k = 4$	...	...	...
$k = 3$	$A[3] > A[2] ? (\dots > \dots ?)$	...	...
$k = 2$	$A[2] > A[1] ? (\dots > \dots ?)$	...	$G, \dots, \dots, \dots, J$

Hasil akhir langkah 2:

UNIT	G	B	...	...	...	...	...	...	...	...
JUMLAH	71	398	...	...	...	...	...	...	...	...

3. ...

4. ...

.

.

9. Langkah 9

k	Elemen yang dibandingkan	Pertukaran?	Hasil sementara
$k = 10$	$A[10] > A[9]$ ( $J > I$ tidak)	tidak	$\dots, \dots, \dots, \dots, \underline{I}, J$

Hasil akhir langkah 9:

UNIT	G	B	...	...	...	...	...	...	...	...	...
JUMLAH	71	398	...	...	...	...	...	...	...	...	...

1    2    3    4    5    6    7    8    9    10

Dari langkah ke sembilan baru didapat hasil pengurutan berdasarkan peringkat tertinggi

#### 2.5.2.4. Mengasosiasikan atau menalar

Pada permasalahan 1 memiliki langkah yang lebih sedikit dibandingkan dengan permasalahan menanya. Hal ini disebabkan oleh banyaknya data lebih banyak pada permasalahan menanya. Langkah yang banyak memerlukan waktu yang lebih banyak juga. Sehingga semakin banyak data, maka langkah semakin banyak yang mengakibatkan membutuhkan waktu yang banyak juga.

#### 2.5.3. Rangkuman

Penggunaan algoritma pengurutan gelembung dan seleksi pada aplikasi sangat bermanfaat jika data yang diurutkan hanya sedikit.

#### 2.5.4. Tugas

1. Pada permasalahan 1, data diurutkan menurun (peringkat tinggi kerendah). Maka berdasarkan permasalahan tersebut, urutkanlah secara menaik dengan menggunakan algoritma gelembung.
2. Perhatikan kembali permasalahan 1, lakukan pengurutan dengan menggunakan algortima pengurutan seleksi-minimum, baik menaik maupun menurun.

#### 2.5.5. Uji Kompetensi

A.

1. Teknik dalam menyeleksi, membandingkan, dan memilih sebuah dari beberapa data yang ada disebut?
  - a. Sorting
  - b. Conquer
  - c. Searching
  - d. Devide
2. Pencarian data dari membandingkan data satu per satu dari posisi awal secara beruntun disebut dengan istilah?
  - a. Random searching
  - b. Binary searching
  - c. Binari searching
  - d. Linier/sequential searching

3. Dasar dari algoritma pencarian linier adalah?
  - a. Mencari data
  - b. Membandingkan data satu per satu
  - c. Mengecek data
  - d. Menelusuri data
4. Terdapat suatu array dengan elemen sebanyak 1000, jika kita akan melakukan pencarian dengan menggunakan linier searching, maka waktu maksimal yang dibutuhkan adalah
  - a. 1000 kali
  - b. 999 kali
  - c. 500 kali
  - d. 100 kali
5. Dari beberapa hal berikut, yang mempengaruhi algoritma sorting atau pengurutan adalah
  - a. Jumlah operasi perhitungan
  - b. Jumlah operasi perbandingan dan jumlah operasi pemindahan
  - c. Jumlah operasi perbandingan
  - d. Jumlah operasi pemindahan
6. Kegiatan mengurutkan sekumpulan data dalam array disebut?
  - a. Sorting
  - b. Conquer
  - c. Searching
  - d. Devide
7. Algoritma yang diinspirasi dari gelembung sabung adalah algoritma?
  - a. Linier
  - b. Apung
  - c. Seleksi
  - d. Sisip
8. Misalkan akan mengurutkan sekumpulan data dengan  $n$  banyak data. Jika ingin mengurutkan dengan algoritma apung, berapa langkah yang akan ditempuh?
  - a.  $n$
  - b.  $n - 1$
  - c.  $n - 2$
  - d.  $n - 3$
9. Algoritma pengurutan yang memilih elemen maksimum atau minimum kemudian meletakkannya di awal atau akhir urutan disebut?
  - a. Apung
  - b. *Bubble*
  - c. Sequential
  - d. Seleksi

10. Misalkan akan mengurutkan sekumpulan data dengan  $n$  banyak data. Jika ingin mengurutkan dengan algoritma seleksi, berapa langkah yang akan ditempuh?

- a.  $n - 3$
- b.  $n - 2$
- c.  $n - 1$
- d.  $n$

B.

1. Apa yang dimaksud dengan algoritma pencarian liner? Sebutkan kelemahan dan kelebihannya!
2. Lakukan pencarian angka 100 dengan algoritma linier sentinel array berikut!

89	109	63	21	53	10	
—	1	2	3	4	5	6

3. Sebutkan perbedaan dari algoritma pengurutan apung dan algoritma seleksi! Menurut kamu, algoritma pengurutan mana yang lebih efisien? Beri alasan!
4. Perhatikan data array berikut!

31	15	64	5	64	15	31	
—	1	2	3	4	5	6	7

- a. Lakukan pengurutan dengan algoritma pengurutan gelembung menaik!
- b. Lakukan pengurutan dengan algoritma pengurutan gelembung menurun!
5. Pada data array soal no.4, lakukan:
  - a. Pengurutan dengan algoritma pengurutan seleksi-maksimum menaik!
  - b. Pengurutan dengan algoritma pengurutan seleksi-maksimum menurun!
  - c. Pengurutan dengan algoritma pengurutan seleksi-minimum menaik!
  - d. Pengurutan dengan algoritma pengurutan seleksi-minimum menurun!

Lembar Kreativitas Siswa.

```
program
uses
var

procedure
var
begin

end;

//program utama
begin

end.
```

# BAB

## III

### ***PENGEMBANGAN APLIKASI***

#### Kompetensi Dasar:

- Menerapkan bahasa pemrograman pada aplikasi bisnis
- Memecahkan kasus aplikasi bisnis menggunakan konsep bahasa pemrograman prosedural

## BAB III

### PENGEMBANGAN APLIKASI

Pembahasan pada bab sebelumnya ditujukan agar siswa lebih menguasai konsep pemrograman dengan bantuan bahasa Pascal. Setelah siswa menguasai konsep pemrograman maka pada bab III ini siswa berlatih dan belajar untuk menjadi seorang perekayasa perangkat lunak atau lebih tepatnya *software engineer*. Seorang *engineer* bukan berfokus pada pembuatan program melalui penulisan baris-baris perintah (*source code*) tetapi lebih kepada bagaimana menghasilkan perangkat lunak yang berkualitas dan sesuai dengan kebutuhan atau sesuai dengan keinginan *customer* sebagai pemesan perangkat lunak.

Rekayasa perangkat lunak sendiri sebenarnya adalah suatu disiplin ilmu yang menjadi bagian dari ilmu komputer dan sampai sekarang masih terus dikembangkan. Ilmu rekayasa perangkat lunak berbeda dari ilmu rekayasa produk nyata yang dibuat secara massal dan seragam, misalnya produk pabrikan seperti mobil, televisi, biscuit kaleng, sepatu, dan produk-produk lain yang semacam. Perbedaan ini muncul karena adanya perbedaan karakteristik perangkat lunak dibanding produk lainnya. Karakteristik khusus dari perangkat lunak adalah sebagai berikut.

1. Perangkat lunak tidak pernah usang, meskipun sudah digunakan selama bertahun-tahun. Pengguna perangkat lunak mungkin mengalami kebosanan tetapi sebenarnya kondisi perangkat lunak akan tetap sama seperti awal asalkan kondisi hardware tidak mengalami perubahan.
2. Perangkat lunak bukan produk pabrikan massal (*manufactured product*), tetapi hasil pengembangan dan rekayasa (*developed and engineered product*).
3. Meski industri pada umumnya mengarah pada perakitan berbasis komponen, kebanyakan perangkat lunak masih dibuat berdasar keinginan pemesan (*custom built*).

Saat membuat perangkat lunak, demi menghasilkan perangkat lunak yang berkualitas baik, sangat penting untuk menerapkan sederetan tahap pengembangan perangkat lunak. Sederetan tahap ini disebut dengan proses perangkat lunak. Proses perangkat lunak bertindak sebagai kerangka utama. Untuk menerapkan kerangka utama ini diperlukan suatu strategi yang disebut dengan model proses. Model proses adalah sebuah paradigma atau konsep teknis yang menjadi acuan saat membuat sebuah perangkat lunak.

Ada banyak model proses tetapi ada model proses yang menjadi induk dari semua model proses yang ada. Diantara model proses yang menjadi induk antara lain adalah model air terjun (*waterfall model*) dan model purwarupa (*prototyping model*). Tahapan utama dalam model waterfall adalah *Analysis – Design – Coding – Testing – Support and Maintenance* yang sering disingkat ADCTsM atau ADCTM saja.

### 3.1. Kegiatan Belajar 1. Model Waterfall Tahap Analisis

**Alokasi Waktu : 3 x 45 menit**

#### 3.1.1. Tujuan Pembelajaran

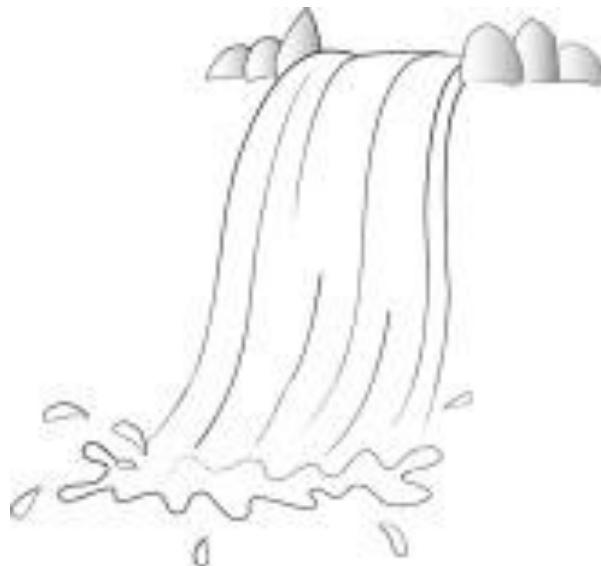
Tujuan pembelajaran pada Kegiatan Belajar 1 Model Waterfall Tahap Analisis adalah:

1. Memahami tahap analisis perangkat lunak.
2. Membuat analisis perangkat lunak.

#### 3.1.2. Aktivitas belajar siswa

##### 3.1.2.1. Mengamati/ observasi

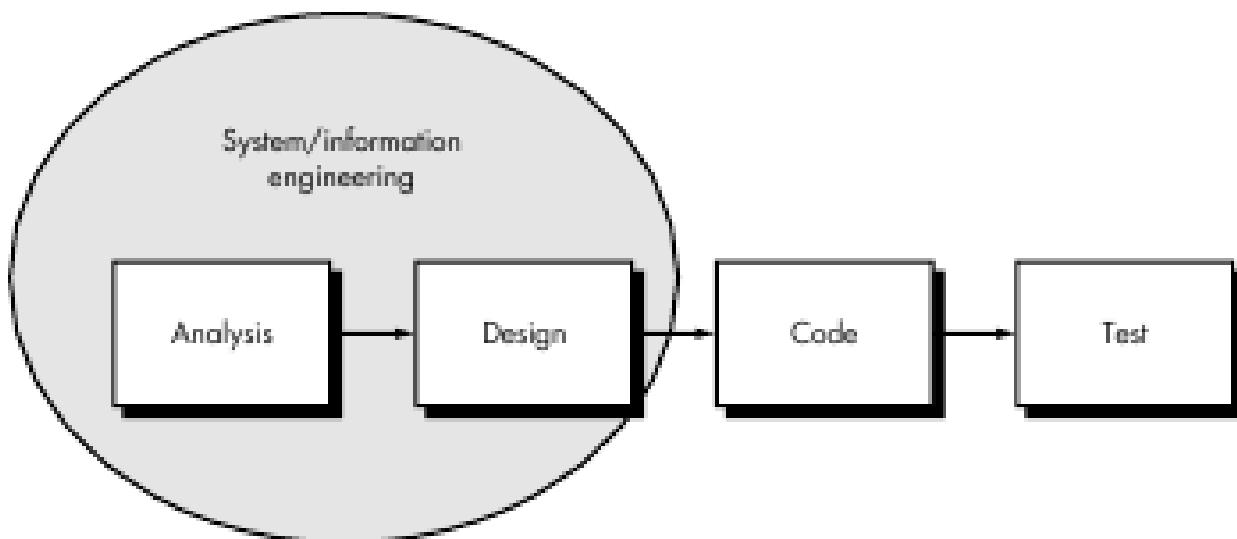
Untuk mengenal tahapan-tahapan dalam model waterfall, perhatikan gambar berikut. Coba pikirkan bagaimana proses pergerakan air terjun.



**Gambar 3.1.** Sketsa air terjun (<http://pixbim.com/4021>)

Ingat pula urutan proses berpikir *scientific* yaitu: mengamati – menanya – mencoba/ mengeksplorasi – menalar/mengasosiasi – menyimpulkan. Ingat juga langkah-langkah pemecahan masalah yaitu: memahami – merencanakan – melaksanakan – memeriksa kembali. Perhatikan bahwa proses tersebut dilakukan secara urut, sama seperti air terjun hanya bergerak dari arah atas ke bawah. Kemudian bagaimana urutan tahap dalam model waterfall? Sama seperti hendak menyelesaikan masalah, maka langkah pertama adalah memahami masalah, kemudian membuat rencana pemecahan masalah.

Urutan dalam model waterfall adalah Analisis – Desain – Pengkodean (coding) – Pengujian (testing) – Dukungan dan Perawatan (support and maintenance). Tahap analisis adalah tahap pengenalan masalah dan setelah masalah dikenali maka ditetapkan tujuan yang akan dicapai dan solusi terbaik yang akan diambil untuk menyelesaikan masalah tersebut.



Gambar 3.2. Skema model waterfall (Software Engineering, Pressman)

Secara rinci dalam tahap analisis perangkat lunak adalah sebagai berikut.

1. Kenali masalah.

Misalnya seorang customer hendak memesan perangkat lunak kepada seorang pengembang perangkat lunak, maka pertama kali harus diketahui ada masalah apa saja yang ingin diselesaikan dengan software, contohnya pekerjaan pembukuan secara manual menghadapi kendala dan diinginkan agar bisa berjalan lebih efektif, efisien, dan akurat sehingga diinginkan ada software yang bisa membantu pekerjaan menjadi lebih mudah. Langkah berikutnya adalah mengetahui bagaimana *business rule* atau aturan kerja secara manual yang sudah diterapkan untuk diperiksa apakah sudah cukup baik atau masih perlu disempurnakan.

2. Tetapkan tujuan

Setelah masalah dikenali maka langkah berikutnya adalah menetapkan tujuan-tujuan dari pembuatan software. Kegiatan ini sering disebut dengan analisis kebutuhan fungsional, yaitu menetapkan fungsi-fungsi pokok software yang seharusnya diimplementasikan, misalnya fungsi tambah data, update data, hapus data, pencarian data, fungsi pembuatan laporan, fungsi cetak laporan atau eksport laporan dalam jenis berkas yang berbeda. Kegiatan ini dilengkapi dengan analisis kebutuhan non fungsional, misalnya menentukan target sistem operasi dimana sofware bisa berjalan baik, menentukan hardware minimal agar software bisa berjalan baik.

3. Tentukan solusi yang akan diwujudkan

Setelah tujuan ditetapkan maka dibuat beberapa alternatif yang akan diambil atau diwujudkan. Misalnya apakah aplikasi akan dibuat berbasis desktop, web, atau mungkin mobile, apakah akan menggunakan server database atau file penyimpanan data lainnya, apakah akan dibuat bisa diakses secara online atau tidak. Dari semua alternatif dipilih solusi terbaik yang sesuai dengan keinginan customer. Tujuan dan rencana solusi yang sudah dihasilkan menjadi kontrak kerja antara pengembang software dengan pemesannya.

4. Buat model analisis

Pada kegiatan sebelumnya, semua disajikan menggunakan kalimat secara naratif deskriptif yang mungkin saja dilengkapi dengan skema maupun tabel. Agar lebih baku maka diperlukan pemodelan sehingga bisa lebih mudah dipahami secara teknis oleh pembuat software. Model yang sering digunakan antara lain diagram alir data (*Data Flow Diagram/DFD*), diagram keterkaitan data (*Entity Relationship Diagram / ERD*) dan kamus data (*Data Dictionary / DD*).

### 5. Dokumentasi.

Semua yang dihasilkan pada kegiatan analisis harus didokumentasikan, dicatat secara rinci dan terstruktur dan akhirnya harus disimpan dengan baik.

Analisis ini dilakukan dengan cara melakukan diskusi antara customer dan pengembang software. Dokumen analisis menjadi kontrak kerja yang sah antara pemesan dan pembuat. Kualitas software sangat ditentukan oleh kualitas analisis sehingga tahap ini harus dilakukan dengan cermat. Sesuai dengan sifat air terjun yang hanya bergerak ke bawah, maka setiap tahap dalam model waterfall secara alami akan berpindah begitu tahap sebelumnya dianggap sudah selesai. Dalam hal ini, jika tahap analisis disetujui bersama suda final maka kegiatan harus segera berpindah pada tahap desain. Perubahan isi pada sebuah tahap sulit dilakukan jika sudah masuk tahap lanjutan. Ibarat membuat rumah, begitu RAB dan desain rumah sudah disetujui maka sulit sekali melakukan modifikasi, jika dilakukan modifikasi pun berimbang pada semakin lamanya waktu yang dibutuhkan dan terutama berimbang pada biaya karena mengubah kontrak kerja.

### Diagram Alir Data (DAD)

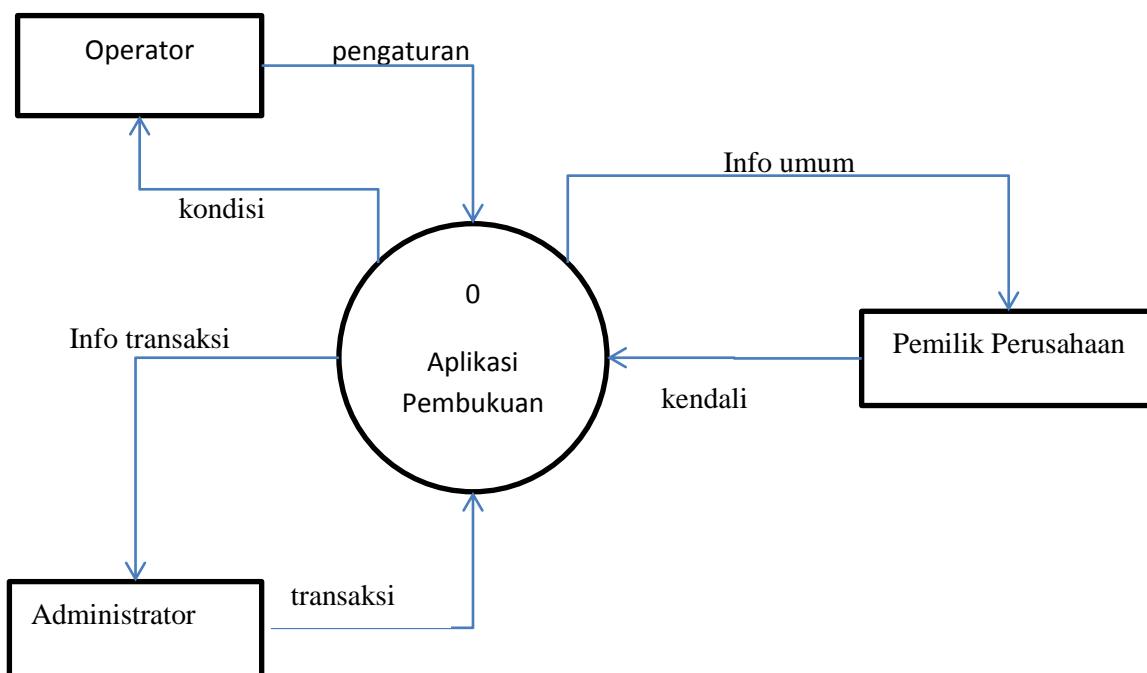
Simbol yang digunakan dalam pembuatan aliran data menurut Pressman adalah sebagai berikut.

Simbol	Bentuk	Kegunaan
Lingkaran		Mewakili suatu proses atau fungsi
kotak persegi panjang		Mewakili entitas luar, misalnya pengguna (user), hardware input/output, atau sistem luar
kotak persegi panjang dengan sisi kiri terbuka		Mewakili media penyimpanan data eksternal, bisa berupa database atau file lainnya
Garis		Mewakili aliran data
ujung panah		Menunjukkan arah aliran data

**Gambar 7.** Simbol DAD

DAD dibuat menjadi beberapa level sesuai dengan kebutuhan. DAD pada level lanjut digunakan untuk memberikan gambaran yang lebih detail dari level sebelumnya. DAD dimulai dari level 0 dan sering disebut dengan diagram alir kontek (*Context Flow Diagram / CFD*). CFD hanya berisi satu lingkaran besar yang mewakili sistem atau software yang akan dibuat dan entitas luar

sistem. CFD tidak memuat simbol data eksternal. Setiap simbol DFD diberi penjelasan. Perhatikan contoh berikut.



Gambar 8. Contoh diagram alir kontek aplikasi multilevel-user

Dari diagram alir kontek dikembangkan DAD level 1. Agar lebih mudah dalam peruntutan dokumen maka penomoran setiap lingkaran/proses bisa dibuat secara terstruktur. Setiap garis harus disertakan nama data yang dialirkkan dan sebuah garis hanya memiliki satu arah saja, tidak boleh memiliki ujung panah pada kedua sisinya. DAD level satu bisa berisi beberapa lingkaran/proses. Dari setiap lingkaran yang ada pada level satu bisa dibuat DAD level 2, sehingga banyaknya DAD level dua maksimum adalah sebanyak lingkaran pada level satu karena tidak setiap lingkaran harus didetailkan lagi jika dirasa sudah cukup jelas. Banyaknya DAD yang harus dibuat akan sejalan dengan tingkat kerumitan suatu software.

### 3.1.2.2. Menanya

Berdasarkan kegiatan mengamati, coba pikirkan bagaimana membuat DAD untuk mengawali pembuatan perangkat lunak. Bisa diambil kasus perangkat lunak untuk pendataan nilai siswa.

### 3.1.2.3. Mencoba

#### Percobaan 1

Mengenali entitas dan fungsi aplikasi.

**Aturan kerja:**

Perangkat lunak dibuat untuk satu pengguna, bukan multilevel user sehingga tidak perlu dilindungi password. Untuk selanjutnya pengguna aplikasi akan disebut dengan user saja. Tujuan utama dari aplikasi adalah untuk melakukan pendataan nilai siswa. Setiap **siswa** akan didata identitasnya dengan cara dicatat nomor induk siswa (NIS), bidang keahlian. Selain itu siswa pasti tercatat pada suatu kelas dan semester tertentu yang bisa berubah seiring jalannya waktu, tetapi NIS dan bidang keahlian tidak boleh berubah. Meskipun aplikasi adalah untuk mendata nilai siswa tetapi siswa tidak bisa menggunakan aplikasi secara langsung dan aplikasi hanya bisa diakses oleh **user operator**. Untuk kelancaran dan kelangkapan maka operator harus **memasukkan** semua mata pelajaran dan kodennya. Aplikasi ini memungkinkan operator untuk **memasukkan** nilai, **mencari** data nilai, **mengubahnya**, dan **menyimpan** data nilai dalam file eksternal.

Coba kenali ada kata benda apa saja dalam aturan kerja tersebut. Kenalilah mana entitas/obyek yang benar-benar terlibat langsung dalam aplikasi.

Coba kenali ada kata kerja apa saja yang perlu diberi perhatian khusus.

Coba perhatikan data apa saja yang digunakan.

### Percobaan 2

Membuat DAD.

Siswa diminta berkelompok. Lengkapilah diagram alir berikut, diskusikan dengan teman satu kelompok.

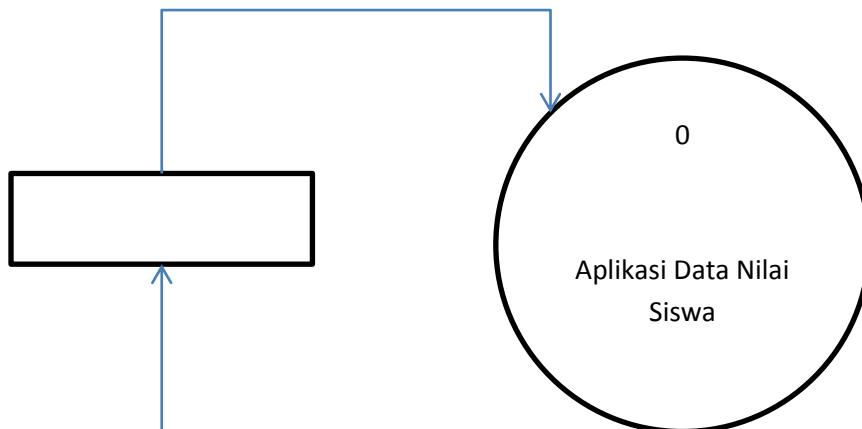
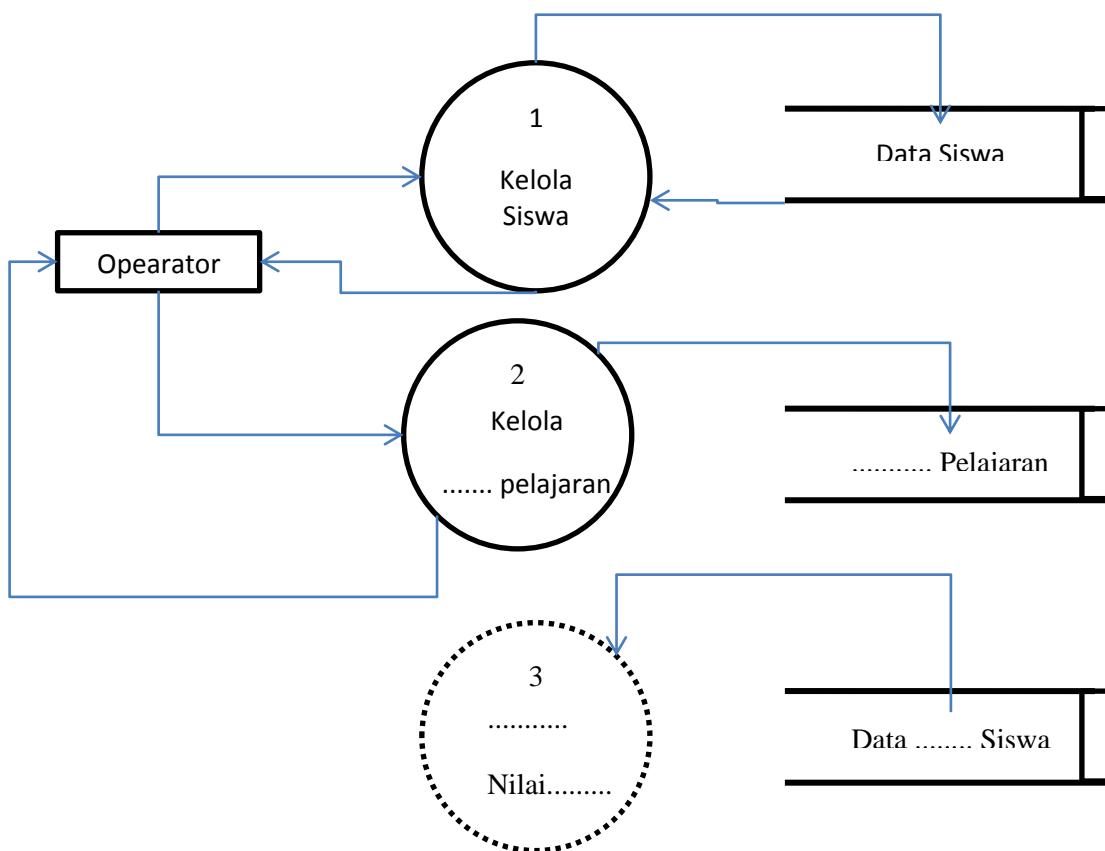


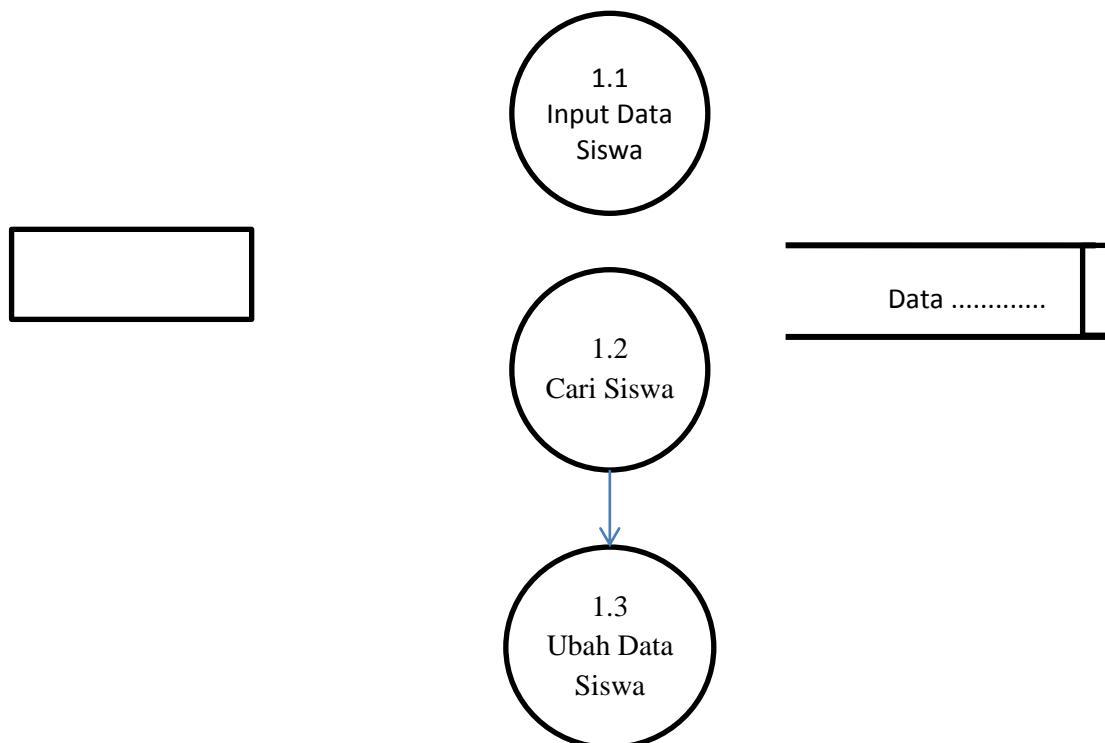
Diagram Konteks

Dari diagram konteks (DAD level 0) coba diskusikan penjabarannya dalam DAD Level 1 berikut.

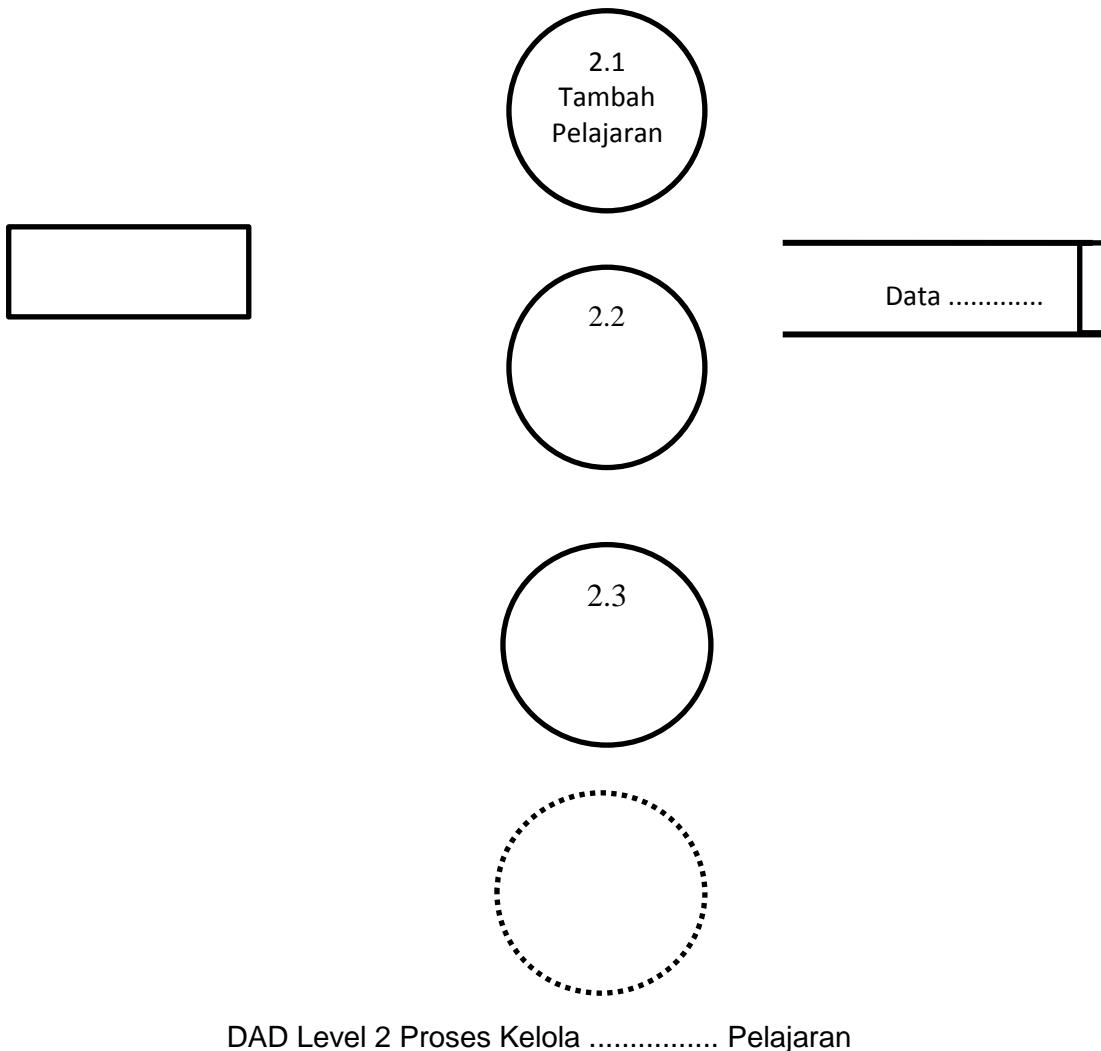


DAD Level 1

Pada DAD Level 1 tersebut ada beberapa lingkaran yang mewakili sebuah fungsi/proses khusus. Tahap berikutnya mari kita perjelas untuk setiap fungsi tersebut ke dalam DAD Level 2.



DAD Level 2 Proses Kelola Siswa



Perhatikan bahwa dari untuk mengubah data seorang siswa pada umumnya akan dicari dulu data siswa tersebut untuk mempermudah. Perhatikan bahwa ada panah dari proses Cari Siswa menuju proses Ubah Data Siswa, artinya proses Ubah Data Siswa bisa didahului dengan pencarian atau proses Cari Siswa bisa dilanjutkan dengan pengubahan data siswa yang ditemukan.

Untuk DAD Level 2 proses yang lain (yang ada di Level 1) silakan didiskusikan dan ditulis dalam buku catatan siswa.

### Percobaan 3

Mengenali data yang dibutuhkan

Nama Data	Atribut
Siswa	NIS, Nama, Bidang keahlian, Kelas, Semester
Pelajaran	Kode, nama pelajaran, .....
Nilai	.....

### 3.1.2.4. Mengasosiasi/ menalar

1. Perhatikan cara mengenali entitas dan data yang dibutuhkan berdasarkan aturan kerja yang dibuat.
2. Perhatikan garis dan arah panah yang dibuat. Coba cari artinya.
3. Perhatikan apakah suatu proses sudah dianggap cukup jelas atau masih harus didetailkan lagi melalui DAD yang lebih tinggi levelnya.
4. Perhatikan cara penomoran setiap lingkarannya.
5. Buatlah kesimpulan dengan berdiskusi.

### 3.1.1. Rangkuman

Dari percobaan-percobaan yang telah dilakukan, bisa ambil kesimpulan bahwa:

1. Analisis dimulai dengan menetapkan aturan kerja dari aplikasi yang akan dibuat.
2. Dari aturan kerja tersebut akan dikenali adanya kata-kata benda yang kemungkinan menjadi entitas luar dan akan dikenali kata-kata kerja yang mungkin menjadi fungsi/proses utama dari aplikasi yang dibuat.
3. DAD dimulai dari diagram konteks kemudian didetailkan melalui DAD level 1, level 2, dan seterusnya sampai dianggap sudah mencukupi.
4. Sebuah proses bisa diawali dari proses lain atau sebuah proses bisa dilanjutkan dengan mengerjakan proses lainnya.
5. Aliran data menuju penyimpanan data (basisdata) hanya berasal dari proses (bukan entitas luar) dan aliran yang keluar dari basisdata hanya menuju proses.
6. Aliran data menuju penyimpanan data (basisdata) diartikan dengan menyimpan data ke basisdata. Aliran data dari penyimpanan data (basisdata) diartika dengan membaca data.

### 3.1.4. Tugas

Buatlah analisis untuk pembuatan aplikasi pendataan nilai yang berlaku di sekolahmu masing-masing.

### 3.1.5. Uji Kompetensi

1. Sebutkan urutan tahap pembuatan perangkat lunak menurut modeel Waterfall.
2. Apa saja yang dilakukan pada tahap analisis?
3. Sebut dan jelaskan setiap simbol yang digunakan dalam pembuatan diagram alir data.
4. Sebutkan perbedaan diagram konteks (level 0) dibanding DAD level satu dan seterusnya?
5. Bagaimana cara menyusun DAD berdasar aturan kerja yang dimiliki?
6. Apakah boleh ada aliran dari sebuah proses ke proses lain? Dalam hal apa yang seperti ini diperbolehkan?

### 3.2. Kegiatan Belajar 2. Model Waterfall Tahap Desain

**Alokasi Waktu : 3 x 45 menit**

#### 3.2.1. Tujuan Pembelajaran

Tujuan pembelajaran pada Kegiatan Belajar 1 Model Waterfall Tahap Desain adalah:

1. Memahami tahap desain perangkat lunak.
2. Membuat desain perangkat lunak.

#### 3.2.2. Aktivitas belajar siswa

##### 3.2.2.1. Mengamati/ observasi

Saat orang hendak membangun rumah kebanyakan dari mereka memuali dari gambar rumah yang diinginkan, gambar itu ada yang dibuat sederhana saja yang mungkin hanya dipahami oleh si pembuat gambar atau mungkin saja gambar teknis yang dibuat oleh arsitek sehingga tentu saja gambar yang dibuat oleh arsitek lebih bisa dipahami oleh pelaksana pembangunan rumah.

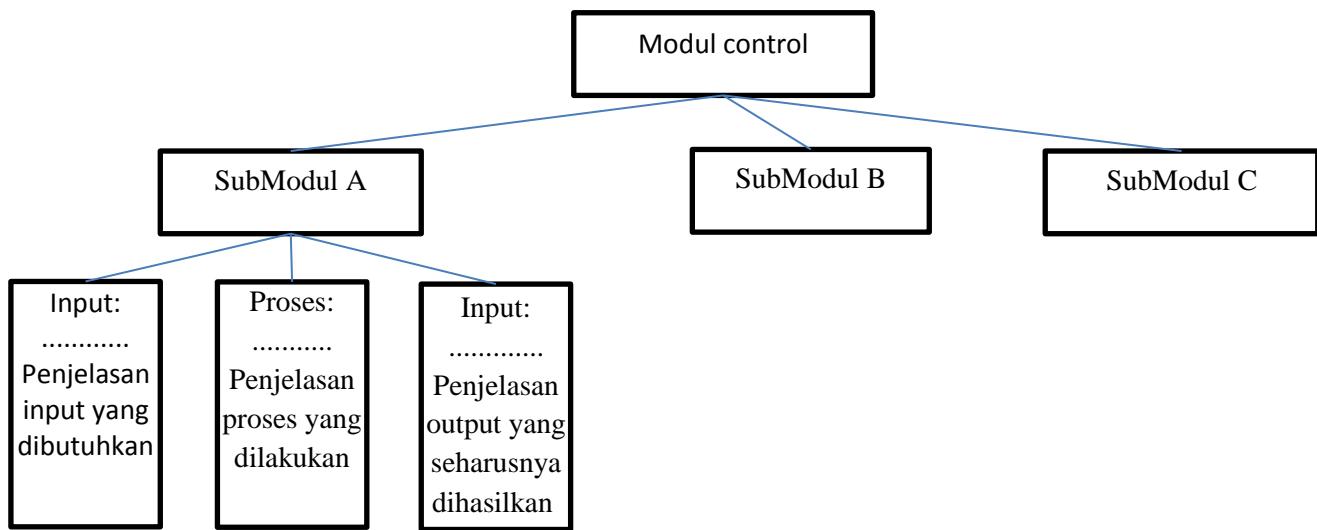
Dalam kasus yang lain sebuah gambar rancangan atau sering disebut dengan desain juga diperlukan saat membuat mobil, peralatan elektronik, dan produk-produk lainnya. Desain akan memberikan pemahaman yang lebih baik bagi pelaksana daripada hanya sekedar dokument-dokumen analisis yang isinya masih sangat global dan bisa diinterpretasikan berbeda oleh pelaksana.

Dalam dunia pembuatan perangkat lunak, desain juga memegang peranan penting karena bisa memberikan gambaran yang lebih jelas bagi programmer untuk mewujudkan aplikasi yang diinginkan. Programmer tidak perlu langsung mengetahui isi dokumen analisis tetapi cukup berpegang pada desain perangkat lunak yang dihasilkan. Oleh karena itu desain perangkat lunak seharusnya bisa dipahami dengan mudah oleh programmer, isinya bersifat teknis pemrograman dan tidak perlu berbelit-belit.

Standar dalam desain perangkat lunak ada banyak, tetapi yang paling sering digunakan adalah desain data, desain arsitektur, dan desain antarmuka. Desain data sendiri adalah penjabaran dari analisis data yang telah dilakukan pada tahap analisis. Desain arsitektur dikembangkan dari diagram alir data (DAD). Desain antarmuka yang paling sederhana adalah antarmuka aplikasi dengan penggunanya, yaitu tampilan aplikasi yang terlihat oleh pengguna.

Simbol yang digunakan dalam desain arsitektur hanyalah kotak dan garis penghubung tanpa disertai arah panah. Secara umum desain arsitektur dan HIPO akan dibaca dari bawah kiri, naik ke atas, ke kanan, kemudian ke kanan bawah.

Desain arsitektur memperlihatkan komponen atau modul atau mungkin hanya prosedur/fungsi yang harus dibuat oleh programmer serta kaitan modul satu dengan modul lainnya. Desain arsitektur juga bisa dilihat sebagai Hierarki Input Proses dan Output (HIPO) yang berlaku pada modul-modul yang harus dibuat. Meski desain arsitektur tidak harus selalu menggunakan HIPO tetapi seringkali hal ini memperjelas programmer sehingga banyak programmer menyukai HIPO. Tetapi HIPO yang baik adalah HIPO yang secara rinci menjelaskan Input yang dibutuhkan agar modul berkerja, menjelaskan tahapan Proses yang dilakukan, dan menjelaskan Output yang seharusnya dihasilkan.

**Gambar 3.5.** Skema desain arsitektur

Desain arsitektur bisa dimulai dari DAD level 1 karena pada DAD level 1 sudah terlihat subproses yang ada. DAD level berikutnya digunakan untuk mengetahui sub dari subproses yang ada.

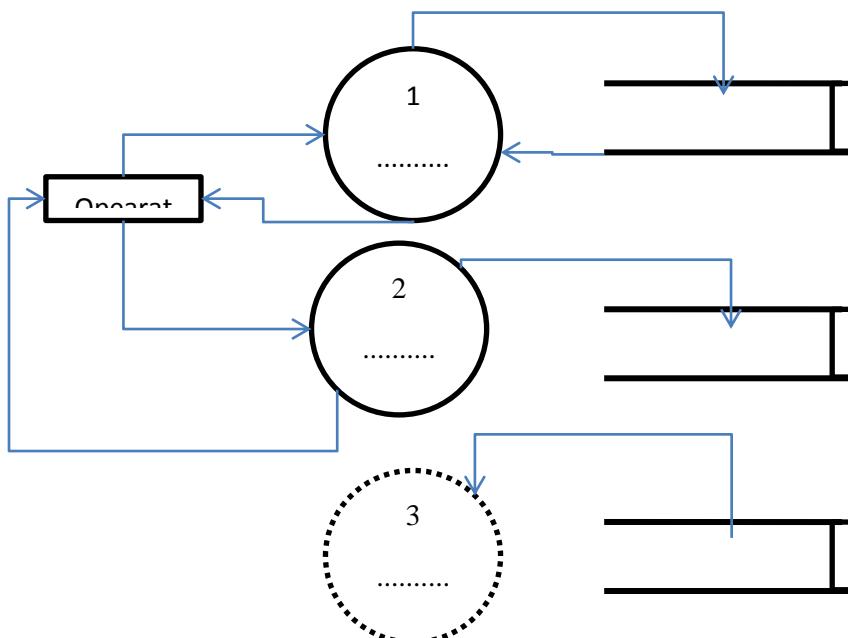
### 3.2.2.2. Menanya

Berdasarkan kegiatan mengamati, coba pikirkan bagaimana membuat desain untuk pembuatan perangkat lunak. Bisa diambil kasus perangkat lunak untuk pendataan nilai siswa berdasarkan analisis pada kegiatan belajar sebelumnya.

### 3.2.2.3. Mencoba

#### Percobaan 1

Lengkapilah skema berikut.

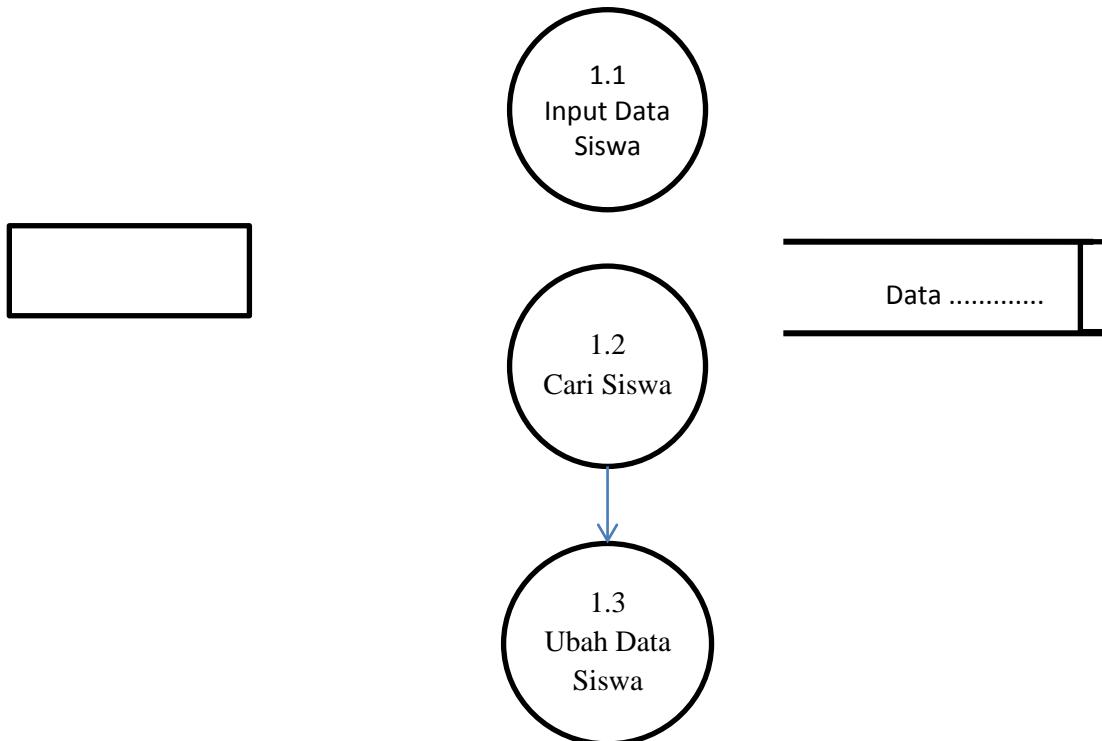


Berdasar DAD Level 1 Tersebut dibuat Desain Arsitektur yang paling dasar. Bisa dilihat ada 3 sub proses utama.

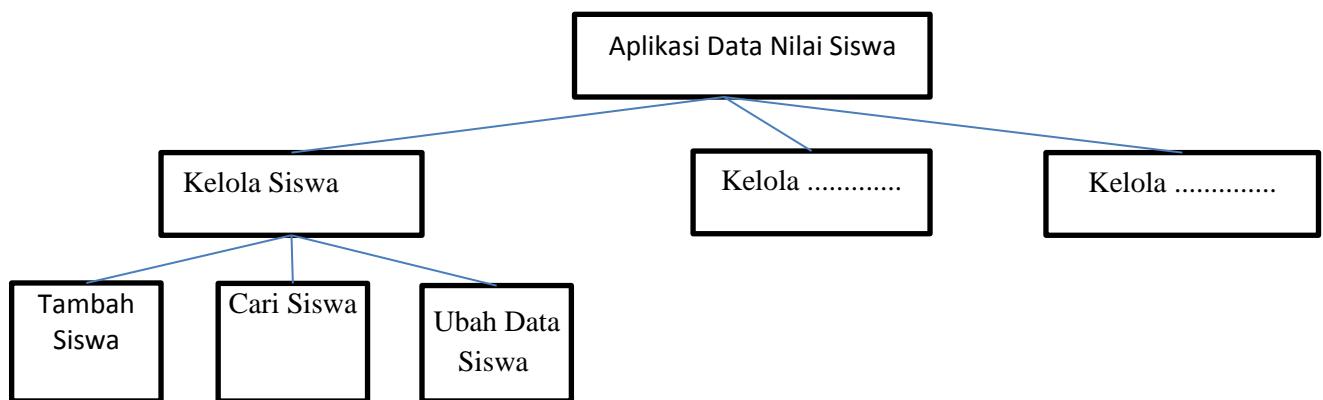


Desain arsitektur awal ini memperlihatkan bahwa programmer harus membuat tiga modul utama untuk mengimplementasikan tiga fungsi utama yang ada dalam aplikasi. Modul yang dibuat bisa berupa fungsi/prosedur, library, atau unit yang sudah dibahas pada Bab I tentang Prosedur dan Fungsi.

Selanjutnya perhatikan DAD Level 2 untuk proses Kelola Siswa. (lengkapi bagian yang belum ada)



Desain arsitektur pada tahap awal kemudian diperbaiki. (Lengkapi bagian yang belum ada)



Desain arsitektur yang dihasilkan sudah lebih detail tetapi belum saatnya untuk menambahkan unsur IPO (Input, Proses, dan Output). IPO baru akan ditambahkan setelah sebuah modul dianggap sudah tidak mempunyai submodul lagi. Desain arsitektur bisa dibuat melebihi DAD yang ada bila dianggap modifikasi tersebut lebih mudah dipahami oleh programmer.

#### 3.2.2.4. Mengasosiasi/ menalar

1. Perhatikan cara membuat dan mendetaIlkan desain arsitektur berdasarkan DAD yang dimiliki.
2. Perhatikan garis-garis penghubungnya.
3. Perhatikan apakah dalam desain arsitektur melibatkan database.
4. Perhatikan apakah dalam desain arsitektur ada penomoran.
5. Perhatikan kaitan antara level DAD dengan tingkat pada desain arsitektur.

#### 3.2.3. Rangkuman

Dari percobaan-percobaan yang telah dilakukan, bisa ambil kesimpulan bahwa:

1. Desain arsitektur dikembangkan dari DAD tetapi boleh dimodifikasi jika memang membuat lebih jelas.
2. Desain arsitektur hanya menggunakan simbol kotak dan garis penghubung tanpa arah panah.
3. Desain arsitektur tidak memperlihatkan database yang digunakan.
4. Desain arsitektur tidak menggunakan sistem penomoran, tetapi ada keterkaitan antara level DAD dengan tingkat hirarki. Proses pada diagram konteks akan berada pada tingkat paling atas, proses pada DAD level 1 akan berada pada tingkat 1 dibawahnya, demikian seterusnya.
5. IPO akan muncul saat sebuah modul tidak memiliki sub modul lagi, sehingga modul pada tingkat paling bawah seharusnya diperjelas dengan IPO.

#### 3.2.4. Tugas

Buatlah desain arsitektur untuk pembuatan aplikasi pendataan nilai yang berlaku di sekolahmu masing-masing.

### 3.3. Kegiatan Belajar 3. Model Waterfall Tahap Pengujian

**Alokasi Waktu : 2 x 45 menit**

#### 3.3.1. Tujuan Pembelajaran

Tujuan pembelajaran pada Kegiatan Belajar 1 Model Waterfall Tahap Pengujian adalah:

1. Memahami tahap pengujian perangkat lunak.
2. Mampu melakukan pengujian perangkat lunak.

#### 3.3.2. Aktivitas belajar siswa

##### 3.3.2.1. Mengamati

Setelah sebuah produk dihasilkan oleh sebuah pabrik maka akan dilanjutkan dengan pengujian produk. Pengujian yang dilakukan untuk memberikan *quality assurance* (jaminan kualitas) agar pelanggan mendapatkan barang yang berkualitas baik dan pelanggan tidak menjadi kecewa atau bahkan mengajukan complain. Kekecewaan pelanggan akan berakibat tidak baik terhadap kelangsungan hidup perusahaan.

Dalam pembuatan perangkat lunak, setelah melalui tahap desain dan kemudian ditulis dalam baris-baris perintah (*coding*) sehingga berwujud perangkat lunak yang bisa digunakan. Mengingat bahwa perangkat lunak memiliki karakteristik yang berbeda dari produk pabrikan pada umumnya maka pengujian perangkat lunak dilakukan dengan cara yang berbeda. Meskipun berbeda tetapi pengujian perangkat lunak memiliki tujuan yang sama dengan pengujian produk lain, yaitu untuk menghasilkan perangkat lunak yang berkualitas tinggi, memberikan jaminan kualitas kepada pengguna perangkat lunak.

##### 3.3.2.2. Menanya

Berdasarkan kegiatan mengamati, coba pikirkan bagaimana melakukan pengujian perangkat lunak.

##### 3.3.2.3. Mencoba

###### Percobaan 1

Berdasarkan tahap analisis Aplikasi Pendataan Nilai Siswa diketahui bahwa aplikasi yang dibuat seharusnya memiliki fungsi:

- Kelola Siswa :
  - + sub fungsi tambah siswa
  - + sub fungsi cari siswa
  - + sub fungsi ubah data siswa
- Kelola Mata Pelajaran
  - + sub fungsi tambah mata pelajaran
  - + sub fungsi cari mata pelajaran
  - + sub fungsi ubah mata pelajaran
- Kelola Nilai Siswa

- + sub fungsi Input Nilai siswa
- + sub fungsi Cari nilai siswa
- + sub fungsi Ubah Nilai siswa
- + sub fungsi Eksport Nilai Siswa

. Seperti sudah disebutkan bahwa dokumen analisis menjadi kontrak kerja yang harus dipenuhi oleh pembuat aplikasi (*software developer*) sehingga pengujian paling utama adalah untuk melihat apakah semua tuntutan tersebut sudah diimplementasikan dengan baik pada aplikasi yang dihasilkan.

Diharapkan pada tahap ini aplikasi Pendataan Nilai Siswa sudah dibuat oleh siswa. Perhatikan aplikasi yang sudah dibuat. Untuk menguji aplikasi berdasar tekniknya ada dua

1. Ujilah semua alur program program.

- + Ujilah semua percabangan yang ada menggunakan nilai variabel yang dibuat bisa melalui semua cabang
- + Ujilah semua perulangan yang terjadi
- + Perhatika perubahan nilai variabel selama jalannya pengujian

Teknik ini membutuhkan bahwa penguji mendapatkan langsung sourcecode asli dari aplikasi yang dibuat. Dalam compiler Free Pascal penguji bisa memanfaatkan menu Step Over (F8) dan Trace Into (F7) dan perhatikan jalannya program secara bertahap. Perhatikan perubahan nilai variabel yang terjadi apakah sudah sesuai atau tidak.

Pengujian dengan teknik ini biasa disebut dengan *white-box testing*.

2. Ujilah semua mekanisme input dan catatlah output yang dihasilkan.

Jalankan program tanpa perlu melihat source-code, sehingga pengujian ini sering disebut *black-box testing*. Cobalah semua fungsi apakah berjalan seperti yang dikehendaki. Untuk setiap fungsi yang ada, coba berikan sederetan input yang valid (benar) dan input invalid (tidak benar).

Input yang valid seharusnya bisa diterima dengan baik untuk selanjutnya diproses, dan hasil dari proses juga harus valid (benar).

Input invalid seharusnya tidak mengakibatkan aplikasi menjadi berhenti atau mati, tetapi aplikasi yang baik harus bisa mengenali input yang invalid, memberi tahu pengguna ada kesalahan dalam input, menolak memproses input yang salah, dan memberitahu input yang seharusnya diberikan oleh pengguna, tanpa mengakibatkan aplikasi menjadi berhenti atau bahkan mati dan lebih parah mengakibatkan keseluruhan proses dalam sistem operasi berhenti.

Dari strategi pengujian juga ada dua, yaitu:

1. Ujilah dalam kondisi yang terkendali, lingkungan aplikasi dibuat seoptimal mungkin agar sistem bisa berjalan dengan baik. Pengujian ini disebut dengan *alpha testing*.

2. Ujilah dalam kondisi lingkungan yang tidak bisa dikontrol, dengan cara:
  - Copy aplikasi ke komputer yang berbeda-beda sistem operasi dan hardwarenya, lihat apakah aplikasi masih bisa berjalan dengan baik.
  - Ujilah dengan penguji yang berbeda-beda kemampuannya, lakukan dalam kelompok dan setiap anggota diminta untuk melakukan pengujian.
  - Ujilah dengan meminta bantuan orang lain yang tidak pernah mengetahui kegunaan aplikasi yang dibuat.

Pengujian semacam ini disebut dengan *beta testing*.

#### 3.3.2.4. Mengasosiasi/ menalar

1. Perhatikan perbedaan cara pengujian yang dilakukan.
2. Carilah keunggulan dari setiap cara pengujian yang dialakukan.
3. Pikirkan siapa pengujian terbaik yang diharapkan bisa memberikan informasi yang berguna untuk meningkatkan kualitas aplikasi
4. Coba diskusikan pengujian yang baik adalah pengujian yang bagaimana.
5. Coba diskusikan apakah mungkin membuang semua kesalahan.

#### 3.3.3. Rangkuman

Dari percobaan-percobaan yang telah dilakukan, bisa ambil kesimpulan bahwa:

1. Pengujian digunakan untuk meningkatkan kualitas aplikasi. Secara alami, setiap produk buatan manusia mengandung kesalahan, sehingga pengujian yang berhasil bukanlah pengujian yang tidak bisa menemukan kesalahan aplikasi. Pengujian dikatakan berhasil jika bisa menemukan kesalahan yang sebelumnya belum diketahui oleh pembuatnya. Meskipun pengujian akan diikuti dengan debugging, tetapi pengujian berbeda dengan debugging, hanya saja akan sangat membantu pembuat aplikasi untuk melakukan debugging jika penguji bisa memberi saran perbaikan.
2. Penguji sebaiknya mereka yang bersifat independent/netral dalam bersikap, tidak membela programmer atau pembuat aplikasi yang secara manusiawi kurang suka jika buatannya dikatakan jelek. Seorang penguji harus mampu mendeskripsikan kondisi aplikasi apa adanya, entah cukup baik (sedikit kesalahan) atau buruk (terlalu banyak kesalahan).
3. Seorang penguji yang juga manusia memiliki keterbatasan, sehingga bagaimanapun perangkat lunak diuji, kemungkinan masih memiliki kesalahan tetaplah ada. Pengujian adalah usaha untuk menghasilkan perangkat lunak yang baik, bukan perangkat lunak yang sempurna dan sama sekali bebas kesalahan.
4. *White-box* testing berguna untuk memastikan bahwa aplikasi berjalan berdasar urutan proses yang benar hingga ke detailnya. *Black-box* testing berguna untuk menjamin bahwa fungsi yang ada telah bekerja dengan benar sesuai yang ditentukan.

5. *Alpha testing* berguna untuk meningkatkan performa atau kinerja aplikasi saat berada dalam lingkungan idealnya. *Beta testing* berguna untuk melihat kompatibilitas (kesesuaian) dan stabilitas aplikasi saat berada pada lingkungan (sistem dan hardware) yang berbeda-beda.

#### 3.3.4. Tugas

Buatlah desain arsitektur untuk pembuatan aplikasi pendataan nilai yang berlaku di sekolahmu masing-masing.

#### 3.3.5. Uji Kompetensi

1. Apakah tujuan dari pengujian perangkat lunak?
2. Bagaimana pengujian perangkat lunak dikatakan berhasil?
3. Siapakah penguji terbaik?
4. Bagaimana melakukan *white-box testing* dan apa kegunaannya?
5. Bagaimana melakukan *black-box testing* dan apa kegunaannya?
6. Bagaimana melakukan *alpha testing* dan apa kegunaannya?
7. Bagaimana melakukan *beta testing* dan apa kegunaannya?

### 3.4. Kegiatan Belajar 4. Model Prototyping

**Alokasi Waktu : 2 x 45 menit**

#### 3.4.1. Tujuan Pembelajaran

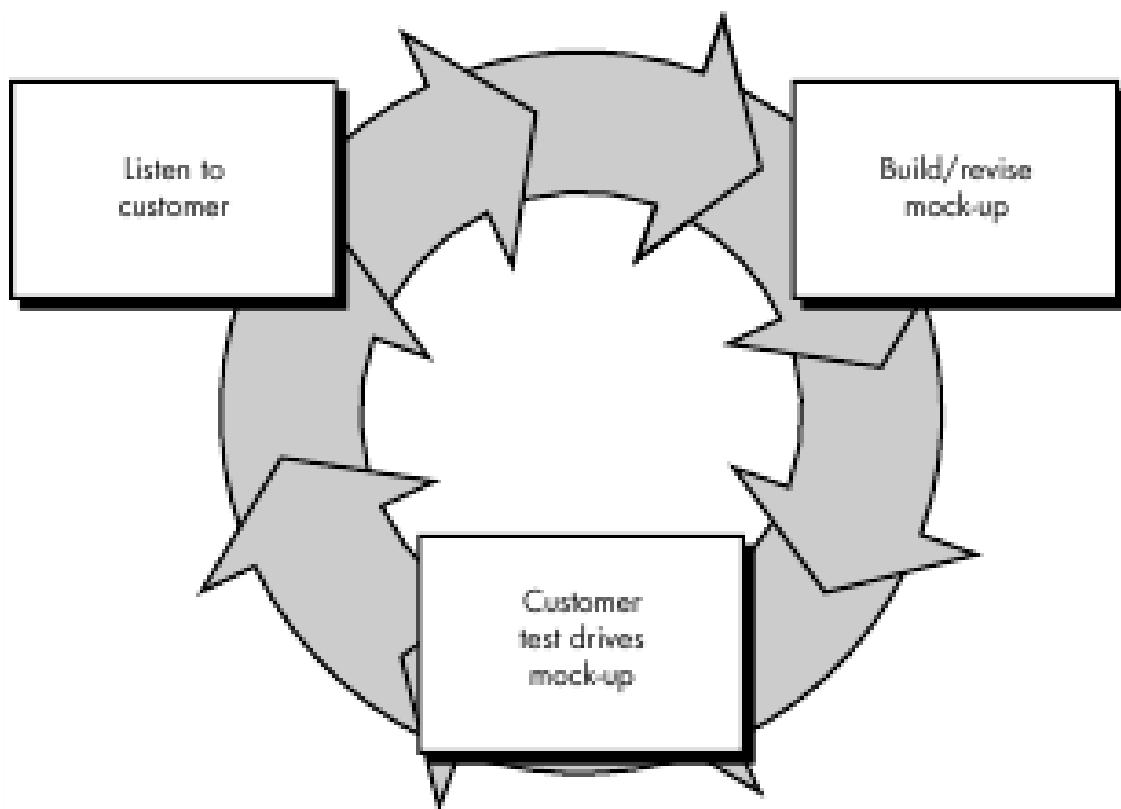
Tujuan pembelajaran pada Kegiatan Belajar 4 Model Prototyping adalah:

1. Memahami model prototyping.
2. Membuat rangcangan perangkat lunak mengikuti model prototyping.

#### 3.4.2. Aktivitas belajar siswa

##### 3.4.2.1. Mengamati

Pada saat seorang gadis berdandang, seringkali diulang-ulang sampai dirasa sudah cantik. Demikian juga saat seorang pengrajin gerabah membuat kerajinan, dia akan selalu mengulang-ulang apa yang dilakukan hingga diperoleh teknik, komposisi bahan, dan model yang bagus. Demikian juga dalam pembuatan perangkat lunak, seringkali pembuat perangkat lunak melakukannya dengan cara seperti itu. Tanpa melalui tahap analisis yang sanat matang dan lama, programmer langsung melakukan pengkodean. Pengkodean itu secara bertahap disempurnakan seiring berjalananya waktu dan seiring pemahaman yang lebih baik tentang apa yang harus dilakukan.



**Gambar 3.6.** Skema prototyping (Software Engineering, Pressman)

Prototyping pada dasarnya melakukan pengembangan perangkat lunak secara cepat, tidak harus mengikuti tahapan ADCT secara matang. Meski tidak terlalu matang tetapi perbaikan bisa

terjadi karena dilakukan secara berulang (siklus). Banyak sekali programmer atau pembuat perangkat lunak menerapkan model ini karena tidak membutuhkan waktu yang terlalu lama hingga aplikasi bisa dicoba. Meskipun demikian, biasanya aplikasi yang dihasilkan pada siklus-siklus awal seringkali sama sekali jauh dari yang diharapkan dan tidak layak digunakan.

Langkah-langkah prototyping adalah sebagai berikut.

1. Dengarkan kebutuhan customer , hal ini pada dasarnya sama seperti tahap analisis hanya saja dilakukan dengan cepat dan tidak perlu sangat detail.
2. Buat prototipe calon aplikasi dengan segera berdasarkan informasi pada tahap satu.
3. Customer menguji prototipe yang dihasilkan. Jika customer sudah merasa cukup maka pengembangan aplikasi dihentikan, jika customer belum puas dan ada informasi tambahan yang lebih detail maka siklus berputar ke tahap satu tetapi diarahkan untuk lebih menyempurnakan aplikasi.

#### 3.4.2.2. Menanya

Berdasarkan kegiatan mengamati, coba pikirkan bagaimana membuat aplikasi dengan menerapkan model prototyping.

#### 3.4.2.3. Mencoba

##### Percobaan 1

Lakukan bersama teman satu kelompok, buatlah ada kelompok pengembang aplikasi (developer) dan kelompok pemesan aplikasi (customer)

1. Tentukan aplikasi apa yang akan dibuat.
2. Jalankan siklus

##### Siklus I

Info Customer Fungsi utama aplikasi	
Prototipe yang dihasilkan	
Uji coba prototipe: Hal Positif dan Negatif	
Keputusan	<input type="checkbox"/> Stop, prototipe sudah bagus <input type="checkbox"/> Prototipe belum bagus, lanjut ke siklus berikutnya

**Siklus II**

Info Customer Fungsi tambahan yang seharusnya ada	
Prototipe yang dihasilkan	
Uji coba prototipe: Hal Positif dan Negatif	
Keputusan	<input type="checkbox"/> Stop, prototipe sudah bagus <input type="checkbox"/> Prototipe belum bagus, lanjut ke siklus berikutnya

Siklus tersebut diulang hingga aplikasi yang dibuat sudah memuaskan pehiaik pemesan (customer).

#### 3.4.2.4. Mengasosiasi

1. Perhatikan mekanisme kerja model prototyping.
2. Cari keunggulan dan kelemahan prototyping.

#### 3.4.3. Rangkuman

Dari percobaan-percobaan yang telah dilakukan, bisa ambil kesimpulan bahwa:

1. Prototyping pada dasarnya pengembangan dari model waterfall, prototyping melakukan tahap ADCT secara cepat dan kualitas aplikasi diperoleh dengan menjalankan aktivitas secara berulang mengarah pada perbaikan.
2. Ada tiga langkah pokok dalam siklus prototyping, yaitu dengarkan kebutuhan customer, buat prototipe dengan segera, uji coba prototipe dengan customer untuk mendapatkan umpan balik.
3. Prototyping memiliki keunggulan bahwa aplikasi sudah mulai terlihat dalam waktu singkat. Kelemahan prototipe adalah karena prosesnya dimana tidak berdasar pemahaman yang menyeluruh dan lengkap sejak awal, sehingga besar kemungkinan setiap fungsi yang ada kurang dipikirkan kaitan satu dengan yang lain, sehingga dimungkinkan aplikasi menjadi sangat berkurang kinerjanya hanya karena sedikit kondisi yang berubah.

### **3.4.4. Tugas**

Lakukan pengembangan aplikasi mengikuti model prototyping.

## **DAFTAR PUSTAKA**

- Cantu, Marco. 2008. *Essential Pascal 4th Edition*. Piacenza Italy: Marco Cantu.
- Kadir, Abdul.1997.*Pemrograman Pascal*.Yogyakarta:Penerbit Andi
- Pressman, Roger S. 2001. *Software Engineering, A Practitioner's Approach, Fifth Edition*. New York: McGraw-Hill.
- Pressman, Roger S. 2012. *Rekayasa Perangkat Lunak, Pendekatan Praktisi Edisi 7 Buku 1* (terjemahan oleh Adi Nugroho, dkk). Yogyakarta: Penerbit Andi.
- Santosa, P. Insap. 1997. *Struktur Data Menggunakan Turbo Pascal 6.0*. Yogyakarta: Andi Offset.
- Suprapto. 2008. *Bahasa Pemrograman Untuk Sekolah Menengah Kejuruan*. Jakarta: DitPSMK Depdiknas
- Van Canneyt, M. 2013. *Free Pascal Programmer's Guide, Programmer's Guide for Free Pascal, Version 2.6.2.* (pdf, freepascal.org)
- Van Canneyt, M. 2013. *Free Pascal Reference Guide, Reference Guide for Free Pascal, Version 2.6.2.* (pdf, freepascal.org)
- Van Canneyt, M. 2013. *Free Pascal User's Guide, User's Guide for Free Pascal, Version 2.6.2.* (pdf, freepascal.org)
- Zarlis, Muhammad dan Handrizal.2008. *Algoritma & Pemrograman: Teori dan Praktik dalam Pascal*. Medan: USU Press
- \_\_\_\_\_. *Pascal Tutorial*. (pdf, tutorialspoint.com)



# Pemrograman Dasar

## SMK/MAK Kelas XI

### Semester 2

Buku Pemrograman Dasar ini ditujukan untuk SMK/MAK bidang keahlian Teknologi Informasi dan Komunikasi kelas XI semester 2.

Buku ini dirancang untuk menjadi buku pegangan siswa agar bisa digunakan untuk membantu pembelajaran yang menerapkan Kurikulum 2013 (K13).

Bahasa pemrograman Pascal dipilih untuk digunakan dalam buku ini karena kesederhaannya tetapi sangat ampuh untuk digunakan untuk mengenal dan menguasai dasar-dasar pemrograman. Compiler yang digunakan dalam penyusunan buku adalah Free Pascal dan menggunakan Integrated Development Environment (IDE) non visual.

Materi:

- Bab I. Prosedur dan Fungsi
- Bab II. Pencarian dan Pengurutan Data
- Bab III. Pengembangan Aplikasi

Meskipun hanya terdiri dari tiga bab tetapi kandungan materi yang dibahas dalam buku sangat banyak, terutama pada Bab I tentang Prosedur dan Fungsi

Buku ini menyajikan beberapa pertanyaan pancingan dan disediakan kegiatan percobaan yang bisa dilakukan siswa, sehingga siswa diajak untuk melakukan percobaan langsung untuk mengetahui jawaban dari setiap pertanyaan yang ada.