# PEMBUATAN LAPORAN TUGAS PRAKTIKUM
# ALGORITMA PEMROGRAMAN I

**Dosen Pengampu : Idhawati Hestiningsih, M.Kom**

Laporan Tugas Praktikum dibuat diakhir praktikum dan dikumpulkan paling lambat saat pertemuan terakhir (minggu ke-12).

Laporan yang dibuat harus mengandung hal-hal sebagai berikut (sistematika penulisan laporan resmi) :

- Cover luar
- Kata Pengantar
- Daftar Isi
- Landasan teori (dari materi praktikum minggu ke-1 s/d minggu terakhir (± minggu ke-10)).
- Permasalahan (dari tugas tiap praktikum)
- Pembahasan masalah terdiri dari
    - Pemecahan masalah
    - Algoritma
    - Flow chart
    - Keterangan variabel yang digunakan
    - Listing program
    - Output program
- Kesimpulan
- Daftar Pustaka

Laporan diketik rapi dan dijilid.

*.....sesungguhnya untuk berhasil di bidang programming tidak diperlukan IQ yang tinggi, melainkan sejauh mana jam terbang kita dalam berlatih memprogram komputer alias "Ngoprek" !*

Kalimat mutiara dari Drs.Bambang Nurcahyo Prastowo, M.Sc.

**Selamat belajar & berlatih, sukses untuk Anda !**

***PRAKTIKUM MINGGU 1***
**PENGENALAN PROGRAM C**
**STRUKTUR RUNTUNAN (SEQUENCE)**

**LANGKAH-LANGKAH PEMBUATAN PROGRAM**

1. Untuk menulis program baru caranya tekan tombol ALT+ F pilih NEW terlebih dahulu sebelum Anda mengetik program Anda.
2. Ketikkan program Anda.
3. Simpan program Anda dengan cara tekan tombol F2 atau ALT + F pilih Save kemudian ketikkan nama program Anda, misalnya `Contoh1.c` <enter>
4. Jalankan program dengan menekan tombol ALT + R pilih RUN atau CTRL + F9
5. Bila ada kesalahan perbaiki kesalahan tersebut kemudian Anda ulangi langkah 3 dan 4
6. Untuk membuat program baru ulangi langkah 1.
7. Bila Anda ingin mencek apakah program Anda sudah tersimpan atau ingin menampilkan program Anda kembali ke layar, maka tekan tombol ALT + F pilih LOAD atau OPEN kemudian enter.
8. Jangan lupa, sebelumnya set isi menu OPTIONS > DIRECTORIES sesuai lokasi directory Turbo C dan file program Anda.

Selanjutnya Anda kerjakan contoh-contoh program dibawah ini beserta tugasnya.

---

**1. BARIS KOMENTAR**

Baris komentar adalah baris-baris yang menjelaskan maksud dari perubah yang digunakan atau maksud dari program itu sendiri. Hal ini dimaksudkan untuk memudahkan pelacakan atas perubah yang digunakan apabila program yang digunakan cukup besar atau memudahkan orang lain memahami program yang kita buat. Dalam program, baris komentar diletakkan diantara tanda /* dan */ dan baris ini tidak dikerjakan oleh komputer, hanya dianggap sebagai baris kosong.

**2. BENTUK / STRUKTUR PROGRAM C**

Bentuk program C mirip dengan kebanyakan program bahasa tingkat tinggi lainnya. Bentuk programnya adalah :

Judul Program
Daftar Header File
Deklarasi
Deskripsi

**Judul Program**

Judul program sifatnya sebagai dokumentasi saja, tidak signifikan terhadap proses program. Ditulis dalam bentuk baris komentar.

Contoh :

```
/*  Program Menghitung Rata-Rata */
```

**2.2 Header File**

C menyediakan sejumlah file judul (header file) yaitu file yang umumnya berisi prototipe fungsi, definisi makro, variabel dan definisi tipe. File ini mempunyai ciri yaitu namanya diakhiri dengan extension .h.

Contoh :

```
#include <stdio.h>
```

Keterangan : menyatakan bahwa agar membaca file bernama stdio.h saat pelaksanaan kompilasi.

**2.3 Deklarasi**

Deklarasi adalah bagian untuk mendefinisikan semua nama yang dipakai dalam program. Nama tersebut dapat berupa nama tetapan (konstanta), nama variabel, nama tipe, nama prosedur, nama fungsi.

**2.4 Deskripsi**

Bagian inti dari suatu program yang berisi uraian langkah-langkah penyelesaian masalah. Program C pada hakekatnya tersusun atas sejumlah blok fungsi. Sebuah program minimal mengandung

sebuah fungsi. Setiap fungsi terdiri dari satu atau beberapa pernyataan, yang secara keseluruhan dimaksudkan untuk melaksanakan tugas khusus.
Bagian pernyataan fungsi (disebut tubuh fungsi) diawali dengan tanda { dan diakhiri dengan tanda }

**3. VARIABEL**

Variabel dalam program digunakan untuk menyimpan suatu nilai tertentu dimana nilai tersebut dapat berubah-ubah. Setiap variabel mempunyai tipe dan hanya data yang bertipe sama dengan tipe variabel yang dapat disimpan di dalam variabel tersebut. Setiap variabel mempunyai nama. Pemisahan antar variabel dilakukan dengan memberikan tanda koma.
Contoh :

```
int jumlah;
float harga_per_unit, total_biaya;
```

Dari contoh diatas,variabel `jumlah` hanya boleh menerima data yang bertipe integer (bulat), tidak boleh menerima data bertipe lainnya. Variabel `harga_per_unit` dan `total_biaya` hanya bisa diisi dengan bilangan float (pecahan).

**4. KONSTANTA**

Berbeda dengan variabel yang isinya bisa berubah selama eksekusi program berlangsung, nilai suatu konstanta tidak bisa berubah.
Contoh :

```
const int m = 8;
#define pajak 0.05
```

**5. FUNGSI `main()`**

Fungsi main() harus ada pada program, karena fungsi inilah yang menjadi titik awal dan titik akhir eksekusi program. Tanda { di awal fungsi menyatakan awal tubuh fungsi sekaligus awal eksekusi program, sedangkan tanda } di akhir fungsi merupakan akhir tubuh fungsi dan sekaligus akhir eksekusi program.

**6. FUNGSI `printf()`**

Merupakan fungsi yang digunakan untuk menampilkan data ke layar. Dengan menggunakan fungsi ini, tampilan dapat diatur (diformat) dengan mudah.
Bentuk umum dari fungsi ini :

```
printf(”string kontrol”, argumen1, argumen2, ....);
```

String kontrol dapat berupa keterangan beserta penentu format (seperti %d, %f). Argumen adalah data yang akan ditampilkan, dapat berupa variabel, konstanta, maupun ungkapan.
Contoh :

```
/* Program Satu */
#include <stdio.h>
main()
{
  Printf(”Belajar Pemrograman Komputer”);
}
```

**7. FUNGSI `scanf()`**

Merupakan fungsi yang digunakan untuk menampilkan data yang dimasukkan dari keyboard.

**8. FUNGSI `getch()`**

Merupakan fungsi yang digunakan untuk membaca sebuah karakter yang dimasukkan dari keyboard dan karakter tersebut tidak akan ditampilkan pada layar. Untuk bisa menggunakan fungsi ini file `conio.h` harus disertakan.

**9. FUNGSI `clrscr()`**

Merupakan fungsi yang digunakan untuk membersihkan (menghapus) layar. Untuk bisa menggunakan fungsi ini file `conio.h` harus disertakan.

**CONTOH PERMASALAHAN 1 :**
Menghitung luas dan keliling lingkaran dengan besar jari-jari lingkaran dimasukkan melalui keyboard.

```
/* Program Dua */
/* Menghitung Luas dan Keliling Lingkaran */
```
*Judul program*

```
#include<stdio.h>
#include<conio.h>
```
*Header File yg digunakan*

```
#define phi 3.14
main()
{
    float jari,luas,keliling;
```
*Deklarasi*

```
    clrscr();
    printf(”Masukan jari-jari lingkaran = ”);
    scanf(”%f”, &jari);
    luas=phi*jari*jari;
    keliling=2*phi*jari;
    printf(”Luas lingkaran = %f \n”,luas);
    printf(”Keliling lingkaran = %f \n”,keliling);
    getch();
}
```
*Deskripsi*

**CONTOH PERMASALAHAN 2 :**
Menukarkan dua buah nilai dari dua buah variabel. Misalnya, sebelum pertukaran nilai a=5, nilai b=3, maka setelah pertukaran nilai a=3, nilai b=5.

```
/* Program Tiga */
/* Menukarkan nilai a dengan b */

#include<stdio.h>
#include<conio.h>
main()
{
    int a,b,c;
    clrscr();
    printf(”Program Menukar 2 Buah Nilai \n\n”);
    printf(”Sebelum ditukar \n”);
    printf(”--------------- \n”);
    printf(”Bilangan pertama = ”);
    scanf(”%i”,&a);
    printf(”Bilangan kedua   = ”);
    scanf(”%i”,&b);
    c=a;
    a=b;
    b=c;
    printf(“Setelah ditukar \n”);
    printf(”--------------- \n”);
    printf(”Bilangan pertama = %i\n”,a);
    printf(”Bilangan kedua   = %i\n”,b);
    getch();
}
```

**CONTOH PERMASALAHAN 3 :**
Mengkonversikan total detik menjadi jam menit detik.
Petunjuk : 1 menit = 60 detik
1 jam = 3600 detik

```
/* Program Empat */
/* Mengkonversi total detik menjadi jam menit detik */

#include<stdio.h>
#include<conio.h>
main()
{
    int totdet,jam,sisa,menit,detik;
    clrscr();
    printf(”Program Konversi Detik Menjadi Jam Menit Detik \n”);
    printf(”------------------------------------------- \n”);
    printf(”Masukkan total detik = ”);
    scanf(”%i”,&totdet);
    jam=totdet/3600;
    sisa=totdet%3600;
    menit=sisa/60;
    detik=sisa%60;
    printf(”Jumlah jam = %i jam \n”,jam);
    printf(”Jumlah menit = %i menit \n”,menit);
    printf(”Jumlah detik = %i detik \n”,detik);
    getch();
}
```

## TUGAS

Seorang user di warnet mulai menggunakan internet pada pukul J1 dan selesai pada pukul J2. Bila tarip penggunaan di warnet tersebut 1 jam Rp. 5000,- maka **buat program billing warnet** untuk menghitung lama pemakaian (dalam jam menit detik) dan biaya yang harus dibayar user.

***PRAKTIKUM MINGGU 2***
**PEMILIHAN KONDISI (SELECTION)**

**1. PILIHAN TUNGGAL**

Bentuk paling sederhana pilihan tunggal adalah jika hanya ada satu pilihan kondisi yang disediakan.

```
if  kondisi
{
  true statement
}
```

**CONTOH PERMASALAHAN 4 :**

Menentukan huruf vokal dari suatu huruf yang dimasukkan dari keyboard.

```
/* Program Lima*/
/* Program menentukan huruf vokal */
   #include<stdio.h>
   #include<conio.h>
   main()
   {
     char huruf;
     clrscr();
     printf("Program Menentukan Huruf Vokal \n");
     printf("------------------------------ \n \n");
     printf("Masukkan huruf : ");
     scanf("%c",&huruf);
     if (huruf=='a'||huruf=='i'||huruf=='u'||huruf=='e'||huruf=='o')
        printf("Huruf %c adalah huruf vokal",huruf);
     getch();
    }
```

**2. PILIHAN GANDA**

Digunakan untuk menentukan tindakan yang akan digunakan bila kondisi bernilai benar dan salah.

```
if  kondisi
{
  true statement
}
else
{
  false statement
}
```

**CONTOH PERMASALAHAN 5 :**

Menentukan bilangan terbesar dari dua buah bilangan.

```
  /* Program Enam */
  /* Program menentukan bilangan terbesar dari dua bilangan */
  #include<stdio.h>
  #include<conio.h>
  main()
  {
    int a,b;
    clrscr();
    printf("Masukkan bilangan pertama = ");
    scanf("%i",&a);
    printf("Masukkan bilangan kedua   = ");
    scanf("%i",&b);
    if(a>b)
       printf("Bilangan terbesar adalah %i",a);
    else
       printf("Bilangan terbesar adalah %i",b);
     getch();
  }
```

**3. PILIHAN MAJEMUK**

Untuk menentukan tindakan yang akan digunakan disediakan lebih dari 2 alternatif. Merupakan bentuk statement if dengan statement if lain di dalam if sebelumnya.

```
if  kondisiA
  {
    if kondisiB
      {
        true statementB
      }
    else
      {
        false statementB
      }
   }
else
{
   false statementA
}
```

**CONTOH PERMASALAHAN 6 :**

Mengelompokan nilai dengan ketentuan :
Jika nilai angka >= 90, maka nilai huruf = A
Jika nilai angka >= 80, maka nilai huruf = B
Jika nilai angka >= 70, maka nilai huruf = C
Jika nilai angka >= 60, maka nilai huruf = D
Jika nilai angka < 60, maka nilai huruf = E

```
/* Program Tujuh */
/* Program mengelompokan nilai */
#include<stdio.h>
#include<conio.h>
main()
{
  int nilai;
  clrscr();
  printf("PROGRAM PENGELOMPOKAN NILAI \n");
  printf("-------------------------- \n");
  printf("Masukkan nilai (0 - 100) : ");
  scanf("%i",&nilai);
  if(nilai>=90)
    printf("Nilai = A");
  else
    if(nilai>=80)
      printf("Nilai = B");
    else
      if(nilai>=70)
       printf("Nilai = C");
      else
       if(nilai>=60)
         printf("Nilai = D");
       else
         printf("Nilai = E");
  getch();
 }
```

**4. STRUKTUR CASE (STATEMENT SWITCH)**

Untuk masalah dengan dua pilihan atau lebih, struktur CASE dapat menyederhanakan penulisan IF yang bertingkat-tingkat.

```
Switch(kondisi)
{
case konstanta1 : {Statement-statement ; break}
case konstanta1 : {Statement-statement ; break}
case konstanta1 : {Statement-statement ; break}
case konstanta1 : {Statement-statement ; break}
.......
 }
```

**CONTOH PERMASALAHAN 7 :**

Pemilihan kode jurusan dengan struktur case

```
/* Program Delapan */
/* Program pemilihan kode jurusan */
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
  char nama[15],ket[30],kode;
  clrscr();
  printf("Masukkan nama mahasiswa: ");
  scanf("%s",&nama);
  printf("Pilih kode jurusan [A/B/C/D] : ");
  kode=getche();
  switch (kode)
  {
    case 'A' : {
         strcpy(ket,"Jurusan Teknik Informatika");
         break;
          }
    case 'B' : {
         strcpy(ket,"Jurusan Manajemen Informatika");
         break;
          }
    case 'C' : {
         strcpy(ket,"Jurusan Sistem Informasi");
         break;
          }
    case 'D' : {
         strcpy(ket,"Jurusan Teknik Komputer");
         break;
          }
  }
  printf("\n \n");
  printf("Nama mahasiswa : %s \n",nama);
  printf("Kode jurusan   : %c \n",kode);
  printf("Nama jurusan   : %s \n",ket);
  getch();
}
```

## TUGAS

2. Misalkan karyawan PT. Makmur dikelompokkan berdasarkan golongannya. Upah per jam tiap karyawan bergantung pada golongannya, dengan ketentuan :

| Golongan | Upah per jam |
|---|---|
| A | Rp. 5000,- |
| B | Rp. 7000,- |
| C | Rp. 8000,- |
| D | Rp. 10.000,- |

   Jumlah jam kerja normal selama 1 minggu adalah 48 jam. Kelebihan jam kerja dianggap lembur dengan upah lembur adalah Rp.4000,- per jam untuk semua golongan karyawan.
   Buat program menghitung gaji karyawan mingguan. Data yang dimasukan dari keyboard adalah nama karyawan, golongan, jumlah jam kerja. Data yang dicetak adalah nama karyawan dan gajinya.
   Program dibuat dengan menggunakan struktur IF dan CASE.

3. Diinginkan sebuah program menghitung segitiga siku-siku yang mempunyai tampilan menu dan fasilitas sebagai berikut dan pada setiap pilihan menu, dimasukkan data alas (sisi siku-siku pertama) dan tinggi (sisi siku-siku kedua). Program dibuat dengan menggunakan struktur CASE.

   MENU SEGITIGA SIKU-SIKU

   1. Hitung panjang sisi miring
   2. Hitung luas
   3. Hitung keliling
   4. Keluar program

***PRAKTIKUM MINGGU 3***
**PENGULANGAN (LOOPING)**

**1. STRUKTUR FOR**

Struktur ini digunakan bila kita mengetahui secara pasti banyaknya pengulangan yang akan dilakukan. Pernyataan FOR mempunyai 3 parameter yaitu :

1. Nilai awal (initial value)
2. Test kondisi yang menentukan akhir loop (condition expression)
3. penentu perubahan nilai (incremental expression)

Bentuk FOR :

```
for (initial value; condition expression; incremental expression)
```

Keterangan :

| | | |
|---|---|---|
| Initial value | : | memberikan nilai awal pada variabel kontrol |
| Condition expression | : | ekspresi yang menyatakan berhentinya pengulangan. Jika tes kondisi bernilai salah maka loop akan berhenti. |
| Incremental expression | : | berfungsi menaikkan/menurunkan nilai dari variabel kontrol. Dapat berupa nilai positif (penaikan) / nilai negatif (penurunan) Penaikan : setiap loop operator ++ akan menambah nilai 1 ke variabel kontrol Penurunan : setiap operator -- akan menurunkan nilai 1 pada variabel kontrol |

**CONTOH PERMASALAHAN 8 :**

```
/* Program Sembilan */
/* Program Mencetak Angka Urut (Penaikan) */
#include<stdio.h>
#include<conio.h>
main()
{
  int i;
  clrscr();
  for (i=1;i<=10;i++)
     printf("%d.Hallo,Selamat belajar,Aku yakin Aku pasti bisa...
        \n",i);
getch();
}
```

**CONTOH PERMASALAHAN 9 :**

```
/* Program Sepuluh */
/* Program Mencetak Angka Urut (Penurunan) */
#include<stdio.h>
#include<conio.h>
main()
{
  int i;
  clrscr();
  for (i=10;i>=1;i--)
     printf("%d.Hallo,Selamat belajar,Aku yakin Aku pasti bisa...
        \n",i);
getch();
}
```

**CONTOH PERMASALAHAN 10 :**

```
/* Program Sebelas */
/* Program Mencari Data Terbesar Terkecil */

#include<stdio.h>
#include<conio.h>
main()
{
  int n,i,max,min,bil;
  clrscr();
  printf("Program mencari data terbesar dan terkecil \n\n");
  printf("Masukkan banyaknya data = ");
  scanf("%d",&n);
  printf("Masukkan bilangan ke-1 : ");
  scanf("%d",&bil);
  max=bil;
  min=bil;
  for(i=2;i<=n;i++)
    {
     printf("Masukkan bilangan ke-%d : ",i);
     scanf("%d",&bil);
     if(bil>max)
       max=bil;
     if(bil<min)
       min=bil;
    }
  printf("\n");
  printf("Data terbesar %d \n",max);
  printf("Data terkecil %d \n",min);
  getch();
}
```

**CONTOH PERMASALAHAN 11 :**

```
/* Program Duabelas */
/* Program Mencetak Jumlah Bilangan */

#include<stdio.h>
#include<conio.h>
main()
{
  int i,awal,akhir,jumlah;
  clrscr();
  printf("Masukkan bilangan awal : ");
  scanf("%d",&awal);
  printf("Masukkan bilangan akhir : ");
  scanf("%d",&akhir);
  jumlah=0;
  for (i=awal;i<=akhir;i++)
    {
     jumlah=jumlah+i;
     printf("Jumlah bilangan dari %d sampai %d adalah
        %d",awal,akhir,jumlah);
     }
  getch();
}
```

**CONTOH PERMASALAHAN 12 :**

```
/* Program Tigabelas */
/* Program Mencetak Bilangan Kuadrat, Akar Bilangan dan Jumlahnya */

#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
  int i,kuadrat,jumbil,jumkuadrat;
  float akar,jumakar;
  clrscr();
  printf("TABEL KUADRAT DAN AKAR BILANGAN \n");
  printf("--------------------------------- \n");
  printf("Bilangan   Kuadrat   Akar Bilangan \n");
  printf("--------------------------------- \n");
  jumbil=0;
  jumkuadrat=0;
  jumakar=0;
  for (i=1;i<=10;i++)
  {  kuadrat=i*i;
     akar = sqrt(i);
     jumbil=jumbil+i;
     jumkuadrat=jumkuadrat+kuadrat;
     jumakar = jumakar+akar;
     printf("   %2d        %3d          %7.4f \n",i,kuadrat,akar);
  }
  printf("================================== \n");
  printf("   %2d        %3d          %7.4f
     \n",jumbil,jumkuadrat,jumakar);

  getch();
}
```

## TUGAS

4. Buat progran menghitung jumlah 6 suku pertama dari barisan $1^3$ , $2^3$, $3^3$ , $4^3$ .....
5. Buat program menghitung jumlah dari deret 3,7,11, 15, .....

## 2. STRUKTUR FOR BERSARANG

**CONTOH PERMASALAHAN 13 :**
Membuat tampilan seperti berikut :

```
*****
*****
*****
```

```
/* Program Empatbelas */
/* Program Mencetak Bintang Persegipanjang*/
#include<stdio.h>
#include<conio.h>
main()
{
  int i,j;
  clrscr();
    printf("Contoh Loop Bersarang --> Bintang Persegipanjang \n\n");
  for (i=1; i<=3; i++)
  {
    for (j=1; j<=5; j++)
      {
       printf("*");
      }
    printf(" \n");
  }
  getch();
}
```

*Struktur for bersarang, dimana didalam for ada for*

**CONTOH PERMASALAHAN 14 :**
Membuat tampilan seperti berikut :

```
Masukkan tinggi segitiga : 5
*
**
***
****
*****
```

```
/* Program Limabelas*/
/* Program Mencetak Bintang Segitiga Siku */
#include<stdio.h>
#include<conio.h>
main()
{
  int n,i,j;
  clrscr();
  printf("Contoh Loop Bersarang --> Bintang Segitiga Siku \n\n");
  printf("Masukkan tinggi segitiga : ");
  scanf("%i",&n);
  for (i=1; i<=n; i++)
  {
    for (j=1; j<=i; j++)
      {
       printf("*");
      }
    printf(" \n");
  }
  getch();
}
```

**CONTOH PERMASALAHAN 15 :**
Membuat tampilan seperti berikut :

```
Masukkan tinggi segitiga : 5
*
**
***
****
*****
****
***
**
*
```

```
/* Program Enambelas*/
/* Program Mencetak Bintang Segitiga */

#include<stdio.h>
#include<conio.h>
main()
{
  int n,i,j;
  clrscr();
  printf("Contoh Loop Bersarang --> Bintang Segitiga \n\n");
  printf("Masukkan tinggi segitiga : ");
  scanf("%i",&n);
  printf("\n");
  for (i=1; i<=n; i++)
  {
    for (j=1; j<=i; j++)
      {
       printf("*");
      }
    printf(" \n");
  }
  for (i=n-1; i>=1;i--)
  {
    for (j=1;j<=i;j++)
      {
       printf("*");
      }
    printf(" \n");
  }
  getch();
}
```

## TUGAS

6. Buat program tabel perkalian :

| * | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | . | . | . | . | . | . | . | . | . | . |
| 2 | . | . | . | . | . | . | . | . | . | . |
| 3 | . | . | . | . | . | . | . | . | . | . |
| 4 | . | . | . | . | . | . | . | . | . | . |
| 5 | . | . | . | . | . | . | . | . | . | . |
| 6 | . | . | . | . | . | . | . | . | . | . |
| 7 | . | . | . | . | . | . | . | . | . | . |
| 8 | . | . | . | . | . | . | . | . | . | . |
| 9 | . | . | . | . | . | . | . | . | . | . |
| 10 | . | . | . | . | . | . | . | . | . | . |

7. Buat tampilan seperti berikut :

```
Masukkan  segitiga angka : 5
1
12
123
1234
12345
1234
123
12
1
```

**3. STRUKTUR WHILE**

Struktur ini digunakan bila kita belum mengetahui secara pasti berapakali banyaknya pengulangan yang akan dilakukan. Berakhirnya proses pengulangan ditentukan oleh suatu kondisi. Selama kondisi terpenuhi, maka pengulangan terus dilakukan, dan sebaliknya, bila kondisinya tidak terpenuhi maka pengulangan dihentikan.

Bentuk WHILE :

```
while (condition expression)
  {
    statement-statement;
  }
```

**CONTOH PERMASALAHAN 16 :**

```
/* Program Tujuhbelas*/
/* Program Menghitung Rata-rata Bilangan */

#include<stdio.h>
#include<conio.h>
main()
{
  int n=0;
  float jumlah=0,bil,rata;
  char lagi='Y';
  clrscr();
  while ((lagi=='Y')||(lagi=='y'))
    {
      printf("Masukkan bilangan  : ");
      scanf("%f",&bil);
      jumlah=jumlah+bil;
      n=n+1;
      printf("Apakah Anda akan memasukkan data lagi [Y/T]: ");
      scanf("%s",&lagi);
      printf("\n");
    }
    rata=jumlah/n;
    printf("\n");
    printf("Banyaknya bilangan : %i \n",n);
    printf("Rata-rata bilangan : %.2f \n",rata);
  getch();
  }
```

**3. STRUKTUR DO-WHILE**

DO-WHILE pada dasarnya sama dengan instruksi WHILE. Perbedaannya hanya terletak pada penempatan ekspresi kondisi. Untuk DO-WHILE, kondisi diletakkan pada bagian bawah. Jadi statement-statement yang berada dalam loop akan dikerjakan dahulu baru dilakukan tes terhadap kondisi.

Bentuk DO-WHILE :

```
do
  {
    statement-statement;
  }
 while (condition expression);
```

**CONTOH PERMASALAHAN 17 :**

```
/* Program Delapanbelas*/
/* Program Menghitung Rata-rata Bilangan */

#include<stdio.h>
#include<conio.h>
main()
{
  int n=0;
  float jumlah=0,bil,rata;
  char lagi;
  clrscr();
  do
    {
      printf("Masukkan bilangan  : ");
      scanf("%f",&bil);
      jumlah=jumlah+bil;
      n=n+1;
      printf("Apakah Anda akan memasukkan data lagi [Y/T]: ");
      scanf("%s",&lagi);
      printf("\n");
    }
    while ((lagi=='Y')||(lagi=='y'));
    rata=jumlah/n;
    printf("\n");
    printf("Banyaknya bilangan : %i \n",n);
    printf("Rata-rata bilangan : %.2f \n",rata);
  getch();
  }
```

**CONTOH PERMASALAHAN 18 :**

```
/* Program Sembilanbelas*/
/* Program Membuat Menu Pengulangan */

#include<stdio.h>
#include<conio.h>
main()
{
  int pilih;
  do
    {
      clrscr();
      printf("DAFTAR MENU TOMBO-NGELEH \n");
      printf("------------------------ \n");
      printf("1. Pecel Lele \n");
      printf("2. Tempe Bakar Penyet \n");
      printf("3. Rica-rica Ayam\n");
      printf("4. Bakso Sapi/Kakap \n");
      printf("\n");
      printf("Masukkan pilihan Anda ! (0=selesai) ");
      scanf("%i",&pilih);
      switch(pilih)
      {
     case 1:
       clrscr();
       printf("Harga Pecel Lele Rp.4000,-");
       break;
     case 2:
```

```
      clrscr();
      printf("Harga Tempe Bakar Penyet Rp.3000,-");
      break;
    case 3:
      clrscr();
      printf("Harga Rica-rica Ayam Rp.5000,-");
      break;
    case 4:
      clrscr();
      printf("Harga Bakso Sapi/Kakap Rp.5000,-");
      break;
    case 0:
      clrscr();
      printf("Selesai");
      break;
     }
     getch();
  }
  while (pilih !=0);
}
```

**TUGAS**

8.

***PRAKTIKUM MINGGU 4***
**ARRAY & ANIMASI HURUF**

Array merupakan koleksi data dimana setiap elemen memakai nama dan tipe yang sama serta setiap elemen diakses dengan membedakan indeks arraynya.

Bentuk : `tipe nama_var[ukuran];`

Keterangan :
Tipe : menyatakan jenis elemen array misal int, char, dll
Ukuran : menyatakan jumlah maksimal elemen array

Contoh : int nilai [5]

## 1. ARRAY DIMENSI SATU

### A. ANIMASI HURUF

**CONTOH PERMASALAHAN 19 :**

```
/* Program Duapuluh */
/* Program membuat animasi huruf */

#include<stdio.h>
main()
{
  char huruf[7]="Hore..!";
  int x,y;
  clrscr();
     /* ------menampilkan huruf satu persatu kekanan-------*/
  for (x=0;x<=6;x++)
  { gotoxy(4+x,4);printf("%c",huruf[x]);
    delay(500);
  }
     /* ------huruf bergerak ke kekanan-------*/

  for(x=0;x<=30;x++)
  {
    gotoxy(4+x,4);printf("Hore..!");
    gotoxy(3+x,4);printf(" ");
    delay(10);
  }
     /* ------huruf bergerak ke bawah-------*/

  for(x=0;x<=6;x++)
  {
    for(y=0;y<=20;y++)
    {
     gotoxy(34+x,3+y);printf("      ");
     gotoxy(34+x,4+y);textcolor(5+BLINK);cprintf("Hore..!");
     if (y==20)
       gotoxy(34,4+y);printf("      ");
     delay(50);
    }
  }
getche();
}
```

## B. MENGOLAH DATA DENGAN ARRAY

**CONTOH PERMASALAHAN 20 :**

```
/* Program Duapuluh satu */
/* Program menginput data dengan array */

#include<stdio.h>
main()
{
  char nama[10] [25];
  char nim[10] [25];
  int nilai[10];
  int i,n;
  /* ------- menentukan banyaknya data yg akan diinputkan ---- */
  clrscr();
  printf("Banyak data :");
  scanf("%i",&n);
  printf("\n");
  /* --- input data sesuai dengan banyaknya data yg ditentukan --- */
  for(i=1;i<=n;i++)
    {
    printf("Data ke-%i \n",i);
    printf("Nama  : ");scanf("%s",&nama[i]);
    printf("Nim   : ");scanf("%s",&nim[i]);
    printf("Nilai : ");scanf("%i",&nilai[i]);
    printf("\n");
    }
  /* --- menampilkan data sesuai dengan yg diinputkan --- */
  clrscr();
  gotoxy(10,1);printf("NO   NIM   NAMA     NILAI \n");
  gotoxy(10,2);printf("----------------------------");
  for(i=1;i<=n;i++)
    {
    gotoxy(10,3+i);printf("%2i.",i);
    gotoxy(15,3+i);printf("%s  ",nim[i]);
    gotoxy(20,3+i);printf("%s  ",nama[i]);
    gotoxy(30,3+i);printf("%3i  ",nilai[i]);
    printf("\n");
    }
  getch();
}
```

## 2. ARRAY BERDIMENSI DUA

**CONTOH PERMASALAHAN 21 :**

```
/* Program Duapuluh dua */
/* Program membuat matriks */

#include<stdio.h>
main()
{
int A[100] [100];
int m,n,i,j;

/* ------menentukan banyaknya baris & kolom matriks  ------*/
clrscr();
printf("Matriks berordo m x n \n");
```

```
printf("---------------------- \n\n");
printf("Masukkan banyaknya baris (m) : ");
scanf("%i,%i", &m);
printf("Masukkan banyaknya kolom (n) : ");
scanf("%i,%i", &n);
printf("\n");
/* ------input elemen matriks ------*/
for(i=0;i<m;i++)
  {
  for(j=0;j<n;j++)
    {printf("Elemen matriks A[%i,%i] : ",i+1,j+1);
     scanf("%i",&A[i][j]);
    }
  }
/* ------menampilkan elemen matriks ------*/
  printf("\n");
  printf("Matriks A = \n\n");
  for(i=0;i<m;i++)
    {
    for(j=0;j<n;j++)
    {
    printf("%3i",A[i][j]);
    }
    printf("\n");
    }
  getch();
  }
```

**CONTOH PERMASALAHAN 22 :**

```
/* Program Duapuluh tiga */
/* Program penjumlahan matriks */

#include<stdio.h>
main()
{
int A[100] [100];
int B[100] [100];
int C[100] [100];
int m,n,i,j;

/* ------menentukan banyaknya baris & kolom matriks  ------*/
clrscr();
printf("Matriks berordo m x n \n");
printf("---------------------- \n\n");
printf("Masukkan banyaknya baris (m) : ");
scanf("%i,%i", &m);
printf("Masukkan banyaknya kolom (n) : ");
scanf("%i,%i", &n);
printf("\n");
/* ------input elemen matriks A------*/
for(i=0;i<m;i++)
  {
  for(j=0;j<n;j++)
    {printf("Elemen matriks A[%i,%i] : ",i+1,j+1);
     scanf("%i",&A[i][j]);
    }
  }

/* ------input elemen matriks B------*/
```

```
printf("\n");
for(i=0;i<m;i++)
  {
  for(j=0;j<n;j++)
    {printf("Elemen matriks B[%i,%i] : ",i+1,j+1);
     scanf("%i",&B[i][j]);
    }
  }
/* ------menjumlahkan elemen matriks A dengan B------*/
for(i=0;i<m;i++)
  {
  for(j=0;j<n;j++)
    {
     C[i][j]=A[i][j]+B[i][j];
    }
  }
/* ------menampilkan matriks A------*/
  printf("\n");
  printf("Matriks A = \n\n");
  for(i=0;i<m;i++)
    {
    for(j=0;j<n;j++)
    {
    printf("%3i",A[i][j]);
    }
    printf("\n");
    }
/* ------menampilkan matriks B------*/
  printf("\n");
  printf("Matriks B = \n\n");
  for(i=0;i<m;i++)
    {
    for(j=0;j<n;j++)
    {
    printf("%3i",B[i][j]);
    }
    printf("\n");
    }

/* ------menampilkan matriks C hasil penjumlahan------*/
  printf("\n");
  printf("Matriks C = \n\n");
  for(i=0;i<m;i++)
    {
    for(j=0;j<n;j++)
    {
    printf("%3i",C[i][j]);
    }
    printf("\n");
    }
  getch();
  }
```

***PRAKTIKUM MINGGU 5***
**SUBPROGRAM / FUNGSI**

- Program komputer yang dibuat untuk menyelesaikan permasalahan umumnya berukuran besar.
- Cara terbaik untuk menangani program besar adalah menyusunnya dari potongan-potongan program yang berukuran kecil-kecil (disebut modul) → merupakan konsep dari pemrograman terstruktur yaitu pemrograman yang menitikberatkan pada pemecahan masalah yang kompleks menjadi masalah yang sederhana.
- Program yang terdiri dari modul/subprogram/prosedur/routine lebih mudah ditangani dibanding dengan program yang terdiri dari banyak sekali baris.
- Modul program dalam C disebut fungsi (function)
- Fungsi adalah blok dari kode yang dirancang untuk melakukan tugas khusus.
- Tujuan pembuatan fungsi :
  - program menjadi terstruktur
  - menghemat kode program karena dapat mengurangi duplikasi kode
  - fungsi dapat dipanggil dari program atau fungsi yang lain
  - mempersingkat/memperpendek panjang program
  - mempermudah cek kesalahan

**PENDEKLARASIAN DAN PENDEFINISIAN FUNGSI**

- Fungsi harus dideklarasikan di dalam program pemanggil/program utama, dengan tujuan supaya program pemanggil mengenal nama fungsi tersebut serta cara mengaksesnya.
- Deklarasi fungsi diakhiri ;
- Fungsi didefinisikan dengan diawali tipe data keluaran fungsi (di depan nama fungsi), defaultnya adalah integer.
- Pendefinisian fungsi tidak diakhiri ;
- Aturan pemberian nama fungsi sama dengan aturan penulisan variabel
- Blok fungsi diawali dengan { dan diakhiri dengan }

Bentuk :

```
tipe_data  nama_fungsi (daftar parameter)
```

Keterangan :

daftar parameter : berisi variabel dan tipe variabel yang berfungsi sebagai masukan untuk fungsi tersebut. Masukan tersebut akan diproses untuk menghasilkan nilai tertentu sesuai dengan tipe data fungsi.

contoh : `int tukar (int x, int y)`

**VARIABEL GLOBAL, VARIABEL LOKAL, VARIABEL STATIK**
**VARIABEL GLOBAL**

- Variabel yang dideklarasikan di luar blok fungsi dan bersifat dikenali oleh semua bagian program.
- Data-data yang tersimpan dalam sebuah variabel dapat diakses di setiap blok fungsi
- Disarankan untuk tidak digunakan, karena variabel ini dapat men-sharing-kan data dan dapat diubah secara tidak sengaja oleh suatu blok fungsi, sehingga nilainya bisa berubah.

**VARIABEL LOKAL**

- Variabel yang dideklarasikan dalam suatu blok fungsi tertentu dan hanya dikenal oleh blok fungsi tersebut.
- Variabel lokal akan dihapus dari memori jika proses sudah meninggalkan blok letak variabel lokalnya.

**VARIABEL STATIK**

- Variabel statik sering dipakai sebagai variabel lokal.
- Beda variabel lokal dan variabel statik adalah variabel lokal akan dihapus dari memori jika proses sudah meninggalkan blok letak variabel lokalnya, sedangkan variabel statik akan dihapus dari memori jika program dimatikan.

Contoh :

```
int i,j;        /* variabel global */
main ()
{
  int k,l;      /* variabel lokal */
}

fungsi()
{
  static int m,n;   /* variabel statik */
}
```

**PARAMETER FORMAL & PARAMETER AKTUAL (NYATA)**

- Parameter formal adalah variabel yang ada pada daftar parameter dalam definisi fungsi.
- Parameter aktual (nyata) adalah parameter yang dapat berupa variabel atau konstanta maupun ungkapan yang dipakai dalam pemanggilan fungsi.

Cara melewatkan/mengirim parameter ke dalam fungsi :

- pengiriman parameter dengan nilai (paramater by value) → disebut juga parameter masukan
  - Nilai dari parameter aktual akan disalin ke parameter formal.
  - Nilai parameter aktual tidak dapat berubah sekalipun nilai parameter formal berubah-ubah.

  Contoh :

  A,B parameter aktual
  x, y parameter formal

  A ⟶ x
  B ⟶ y

  Pada saat pemanggilan suatu fungsi, misal :
  A bernilai 20 → x juga bernilai 20
  B bernilai 30 → y juga bernilai 30

- pengiriman parameter secara acuan (parameter by reference) → disebut juga parameter masukan/keluaran
  - perubahan – perubahan yang terjadi pada nilai parameter formal di fungsi akan mempengaruhi nilai parameter aktual.
  - merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi
  - dipakai untuk mengubah isi suatu varabel di luar fungsi dengan pelaksanaan pengubahan dilakukan di dalam fungsi

**FUNGSI YANG MENGEMBALIKAN NILAI**

- Pada dasarnya semua fungsi mengembalikan nilai, tetapi untuk fungsi yang tidak mengembalikan nilai disebut fungsi bertipe **void**.
- Untuk mengembalikan nilai sebuah fungsi, digunakan kata **return** yang diikuti dengan nilai yang akan dikembalikan.
- Dalam sebuah fungsi return bisa mengembalikan beberapa nilai, tetapi setiap kata return hanya bisa mengembalikan sebuah nilai saja.

**CONTOH PERMASALAHAN 23 :** menggunakan fungsi yang tidak mengembalikan nilai

```
Algoritma fungsi_garis
Deklarasi
    function garis () → integer
Deskripsi
     garis()
     write("NO  NIM  NAMA  NILAI")
     garis()


function garis() → integer
Deklarasi
Deskripsi
     write("==================")
```

```
Output program :

==================
NO  NIM  NAMA  NILAI
==================
```

```
/* Program Duapuluh empat */
/* Program sederhana menggunakan fungsi */
/* yang tidak mengembalikan nilai */

#include<stdio.h>
#include<conio.h>

void garis();          /* deklarasi fungsi */
main()
{
  clrscr();
  garis();
  printf("NO   NIM   NAMA     NILAI \n");
  garis();
  getch();
}
void garis()        /* definisi fungsi */
{
  printf("========================== \n");
}
```

*program utama*

*memanggil fungsi garis*

**CONTOH PERMASALAHAN 24 :** menggunakan fungsi dengan parameter by value

```
Algoritma tukar
Deklarasi
    a,b : integer
    function tukar(input x,y : integer) → integer
Deskripsi
     a ← 80
     b ← 11
     write(a,b)
     tukar(a,b)
     write(a,b)

function tukar(input x,y : integer) → integer
Deklarasi
    z : integer
Deskripsi
    z = px
    px = py
    py = z
    write(px,py)
```

*program utama*

*deklarasi fungsi*

*memanggil fungsi tukar,*
*variabel a,b merupakan paramater aktual*

*definisi fungsi*

*variabel x,y merupakan paramater formal by value*
*sehingga nilai x = a, y = b*

```
/* Program Duapuluh lima */
/* Program tukar menggunakan fungsi */
/* dengan parameter by value */

#include<stdio.h>
#include<conio.h>
void tukar(int x,int y); /*deklarasi fungsi*/
main()
{
int a,b;
clrscr();
a=80;
b=11;
printf("Nilai sebelum pemanggilan fungsi \n");
printf("a = %i  b = %i \n",a,b);
tukar(a,b);
printf("Nilai setelah pemanggilan fungsi \n");
printf("a = %i  b = %i \n",a,b);
getch();
}

void tukar(int px,int py)   /* definisi fungsi */
{
int z;
z=px;
px=py;
py=z;
printf("Nilai diakhir fungsi \n");
printf("px = %i py = %i \n",px,py);
}
```

Output program :

Nilai sebelum pemanggilan fungsi
a = 80
b = 11
Nilai di akhir fungsi
px = 11
py = 80
Nilai setelah pemanggilan fungsi
a = 80
b = 11

**CONTOH PERMASALAHAN 25 :** menggunakan fungsi dengan parameter by reference → dibahas lebih mendalam pada Algoritma Pemrograman II

Algoritma tukar
Deklarasi
    a,b : integer
    function tukar(input *px,*py : integer)
Deskripsi
    a ← 80
    b ← 11
    write(a,b)
    tukar(a,b)
    write(a,b)

function tukar(input *px,*py : integer)
Deklarasi
    z : integer
Deskripsi
    z = *px
    *px = *py
    *py = z
    write(*px,*py)

```
/* Program Duapuluh enam */
/* Program tukar menggunakan fungsi */
/* dengan parameter by reference */

#include<stdio.h>
#include<conio.h>
void tukar(int *px,int *py);
main()
{
int a,b;
clrscr();
a=80;
b=11;
printf("Nilai sebelum pemanggilan fungsi \n");
printf("a = %i  b = %i \n\n",a,b);
tukar(&a,&b);
printf("Nilai setelah pemanggilan fungsi \n");
printf("a = %i  b = %i \n\n",a,b);
getch();
}
void tukar(int *px,int *py)
{
int z;
z=*px;
*px=*py;
*py=z;
printf("Nilai diakhir fungsi \n");
printf("*px = %i *py = %i \n\n",*px,*py);
}
```

Output program :

Nilai sebelum pemanggilan fungsi
a = 80
b = 11
Nilai di akhir fungsi
px = 11
py = 80
Nilai setelah pemanggilan fungsi
a = 11
b = 80

**CONTOH PERMASALAHAN 26 :** penjumlahan menggunakan fungsi yang dapat mengembalikan nilai

Algoritma penjumlahan
Deklarasi
    a,b,c : integer
    function jumlah(input x,y : integer) → integer
Deskripsi
    read(a,b)
    c ← jumlah(a,b)
    write(c)

function jumlah(input x,y : integer) → integer
Deklarasi
    hasil : integer
Deskripsi
    hasil ← x + y
    return hasil

```
/* Program Duapuluh tujuh */
/* Program penjumlahan menggunakan fungsi */
/* yang dapat mengembalikan nilai */

#include<stdio.h>
#include<conio.h>
int jumlah(int x,int y);   /* deklarasi fungsi */

main()
{
 int a,b,c;   /* variabel lokal */
 clrscr();
 printf("A =  ");scanf("%d",&a);
 printf("B =  ");scanf("%d",&b);
 c=jumlah(a,b);
 printf("Jumlah kedua bilangan tersebut = %d",c);
 getch();
}

int jumlah(int x,int y)   /* definisi fungsi */
{
  int hasil;        /* variabel lokal */
  hasil= x+y;
  return (hasil);
}
```

*menunjukkan tipe data keluaran fungsi berbentuk integer*

*memanggil fungsi jumlah, variabel a,b merupakan paramater aktual*

*variabel x,y merupakan paramater formal by value, sehingga nilai x = a, nilai y=b*

*isi dari variabel hasil dibawa / dikembalikan ke tempat fungsi pertama kali dipanggil yaitu di bagian blok main() c=jumlah(a,b)*

Output program :

a = 5
b = 6
c = 11

**CONTOH PERMASALAHAN 27 :** matriks penjumlahan menggunakan fungsi

*tipe data bentukan yaitu tipe data yang dibuat sendiri namanya matrix*

```
#include<stdio.h>
#include<conio.h>

typedef int matrix [10] [10];
matrix A,B,C;
int m; /* banyaknya baris */
int n; /* banyaknya kolom */
void inputmatrik(matrix A,matrix B,int m,int n);
void tampilmatrik(matrix A,matrix B,matrix C,int m,int n);
void jumlahmatrik(matrix A,matrix B,matrix C,int m,int n);

main()
{
/* ------menentukan banyaknya baris & kolom matriks  ------*/
clrscr();
printf("Matriks berordo m x n \n");
printf("---------------------- \n\n");
printf("Masukkan banyaknya baris [1-10] : ");
scanf("%i", &m);
printf("Masukkan banyaknya kolom [1-10] : ");
```

*mendefinisikan tipe data bentukan yaitu matrix yang bertipe array integer*

*A, B, C, m ,n merupakan variabel global*

*deklarasi fungsi*

*program utama*

```
scanf("%i", &n);
printf("\n");

/*---proses input,jumlah, dan tampil matriks menggunakan fungsi ---*/
inputmatrik(A,B,m,n);
jumlahmatrik(A,B,C,m,n);
printf("\n\n");
tampilmatrik(A,B,C,m,n);
getch();
}

void inputmatrik(matrix A,matrix B,int m,int n)
{
int i;  /* indeks baris */
int j;  /* indeks kolom */

/* ------input elemen matriks A ------*/
for(i=1;i<=m;i++)
  {
  for(j=1;j<=n;j++)
    {printf("Elemen matriks A[%i,%i] : ",i,j);
     scanf("%i",&(A[i][j]));
    }
  }
  printf("\n\n");

/* ------input elemen matriks B ------*/
for(i=1;i<=m;i++)
  {
  for(j=1;j<=n;j++)
    {printf("Elemen matriks B[%i,%i] : ",i,j);
     scanf("%i",&(B[i][j]));
    }
  }
}

void jumlahmatrik(matrix A,matrix B,matrix C,int m,int n)
{
  int i;   /* indeks baris */
  int j;   /* indeks kolom */
/* ------menjumlahkan matriks A dengan matriks B------*/
  for(i=1;i<=m;i++)
    {
    for(j=1;j<=n;j++)
      {
       C[i][j]= A[i][j]+B[i][j];
      }
    }
   printf("\n\n");
}

void tampilmatrik(matrix A,matrix B,matrix C,int m,int n)
{
  int i; /* indeks baris */
  int j; /* indeks kolom */

/* ------menampilkan elemen matriks A------*/
  printf("Matriks A = \n\n");

  for(i=1;i<=m;i++)
```

*memanggil fungsi*

*i, j merupakan variabel lokal*
*hanya berlaku di dalam fungsi inputmatrik*

```
    {
    for(j=1;j<=n;j++)
      {
       printf("%3i",(A[i][j]));
      }
    printf("\n");  /* ganti baris */
    }
   printf("\n\n");

  /* ------menampilkan elemen matriks B------*/

  printf("Matriks B = \n\n");
  for(i=1;i<=m;i++)
    {
    for(j=1;j<=n;j++)
      {
       printf("%3i",(B[i][j]));
      }
    printf("\n");  /* ganti baris */
    }
   printf("\n\n");

/* ------menampilkan elemen matriks C------*/

  printf("Matriks C = \n\n");
  for(i=1;i<=m;i++)
    {
    for(j=1;j<=n;j++)
      {
       printf("%3i",(C[i][j]));
      }
    printf("\n");  /* ganti baris */
    }
   printf("\n\n");
}
```

***PRAKTIKUM MINGGU 6***
**PENGURUTAN / SORTING**

- Sorting adalah suatu proses pengurutan data yang sebelumnya disusun secara acak atau tidak teratur menjadi urut dan teratur menurut suatu aturan tertentu.
- Pengurutan terbagi menjadi dua yaitu :
  - ascending (pengurutan naik ) : pengurutan dari karakter/angka kecil ke karakter/angka besar
  - descending (pengurutan turun ) : pengurutan dari karakter/angka besar ke karakter/angka kecil.
- Proses yang terjadi pada pengurutan adalah perbandingan data dan pertukaran data (swap)
- Bermacam-macam metode pengurutan, diantaranya :
  - selection sort
  - bubble sort
  - insertion sort : dibahas pada Algo II
  - quick sort : dibahas pada Algo II
  - merge sort : dibahas pada Algo II

**1. SELECTION SORT**

Proses pengurutan menggunakan selection sort secara ascending :

1. Mencari data terkecil dari data pertama sampai data terakhir, kemudian ditukarkan posisinya dengan data pertama
2. Mencari data terkecil dari data kedua sampai data terakhir, kemudian ditukarkan posisinya dengan data kedua
3. Mencari data terkecil dari data ketiga sampai data terakhir, kemudian ditukarkan posisinya dengan data ketiga
4. Dan seterusnya sampai semua data terurut naik. Bila terdapat n buah data yang akan diurutkan, maka membutuhkan (n – 1) proses / langkah pengurutan, dimana data terakhir yaitu data ke-n tidak perlu diurutkan karena hanya tinggal satu-satunya.

Contoh :

Terdapat 5 buah data yang nilainya belum terurut :

40 50 10 41 90

Data akan diurutkan secara ascending menggunakan selection sort :

Karena ada 5 data maka dibutuhkan (5-1) = 4 proses pengurutan.

*Dalam program **Proses ke-sekian** ditunjukkan dengan i,*
***Data ke-sekian** ditunjukkan dengan j*
*indexmin untuk index pembanding*

**<u>Proses ke-1</u>**

indexmin=1

Data ke-2 : data[indexmin] < data[2] data[1] < data[2] 40 < 50
Data ke-3 : data[indexmin] < data[3] data[1] < data[3] 40 > 10 indexmin=3
Data ke-4 : data[indexmin] < data[4] data[3] < data[4] 10 < 41
Data ke-5 : data[indexmin] < data[5] data[3] < data[5] 10 < 90

Karena data[1] <> data[indexmin], yaitu data[1]<>data[3] maka tukar nilai data ke -1 dan data ke-indexmin (data ke-3). Akhir dari proses ke 1 adalah :

10 50 40 41 90

Selanjutnya kondisi data pada akhir proses pertama ini digunakan sebagai masukan pada proses kedua.

**<u>Proses ke-2</u>**

indexmin=2

Data ke-3 : data[indexmin] < data[3] data[2] < data[3] 50 > 40 indexmin = 3
Data ke-4 : data[indexmin] < data[4] data[3] < data[4] 40 < 41
Data ke-5 : data[indexmin] < data[5] data[3] < data[5] 40 < 90

Karena data[2] <> data[indexmin], yaitu data[2]<>data[3] maka tukar nilai data ke-2 dan data ke-indexmin (data ke-3). Akhir dari proses ke 2 adalah :

10 40 50 41 90

Kondisi data pada akhir proses kedua digunakan sebagai masukan pada proses ketiga.

**Proses ke-3**
indexmin=3
Data ke-4 : data[indexmin] < data[4] data[3] < data[4] 50 > 41 indexmin = 4
Data ke-5 : data[indexmin] < data[5] data[4] < data[5] 41 < 90

Karena data[3] <> data[indexmin], yaitu data[3]<>data[4] maka tukar nilai data ke-3 dan data ke-indexmin (data ke-4). Akhir dari proses ke 3 adalah :
10 40 41 50 90
Kondisi data pada akhir proses ketiga digunakan sebagai masukan pada proses keempat.

**Proses ke-4**
indexmin=4
Data ke-5 : data[indexmin] < data[5] data[4] < data[5] 50 < 90
Akhir dari proses ke 4 adalah :
10 40 41 50 90
Proses selesai

```
/* Program selection sort ascending */
#include<stdio.h>
#include<conio.h>
main()
{
int i,n,temp,indexmin,j;
int data[10];
clrscr();
/* ---memasukan data --- */
printf("Selection sort ascending \n");
printf("Banyak data : ");
scanf("%i",&n);
for(i=1;i<=n;i++)
{
  printf("Data ke-%i : ",i);
  scanf("%i", &data[i]);
}
/* --- proses pengulangan pembandingan data --- */
for(i=1;i<=(n-1);i++)
{
  indexmin=i;
  for(j=i+1;j<=n;j++)
  {
    if (data[indexmin] > data[j])
    indexmin=j;
  }
/* --- proses tukar data (swap)--- */
  if (data[i] != data[indexmin])
  {
    temp= data[i];
    data[i]=data[indexmin];
    data[indexmin]=temp;
  }
}
/* --- proses tampil data setelah urut --- */
printf("Setelah pengurutan \n");
for(i=1; i<=n;i++)
  printf("%i ",data[i]);
getch();
}
```

*membandingkan data*

*variabel penampung sementara*

Output program :

```
Selection sort ascending
Banyak data : 5
Data ke-1 :  40
Data ke-2 :  50
Data ke-3 :  10
Data ke-4 :  41
Data ke-5 :  90
Setelah pengurutan
10  40  41  50  90
```

Untuk selection sort data descending, program hanya diganti pada bagian
`if (data[indexmin] > data[j])` menjadi `if (data[indexmin] < data[j])`

## 2. BUBBLE SORT

Proses yang terjadi pada pengurutan bubble sort adalah selalu membandingkan 2 data yang berdekatan. Secara ascending bila data yang berada di sebelah kanannya bernilai lebih kecil maka dipertukarkan sampai semua data terurut sehingga memunculkan data terbesar di posisi paling akhir.

Contoh :

Terdapat 5 buah data yang nilainya belum terurut :

40 50 10 41 90

Data akan diurutkan secara ascending menggunakan bubble sort :

**Proses 1**

Data 1-2 : data[1] > data[2] 40 < 50
40 50 10 41 90
Data 2-3 : data[2] > data[3] 50 > 10 → ditukar
40 10 50 41 90
Data 3-4 : data[3] > data[4] 50 > 41 → ditukar
40 10 41 50 90
Data 4-5 : data[4] > data[5] 50 < 90
40 10 41 50 90

**Proses 2**

Data 1-2 : data[1] > data[2] 40 > 10 → ditukar
10 40 41 50 90
Data 2-3 : data[2] > data[3] 40 < 41
10 40 41 50 90
Data 3-4 : data[3] > data[4] 41 < 50
10 40 41 50 90

**Proses 3**

Data 1-2 : data[1] > data[2] 10 < 40
10 40 41 50 90
Data 2-3 : data[2] > data[3] 40 < 41
10 40 41 50 90
Data 3-4 : data[3] > data[4] 41 < 50
10 40 41 50 90

**Proses 4**

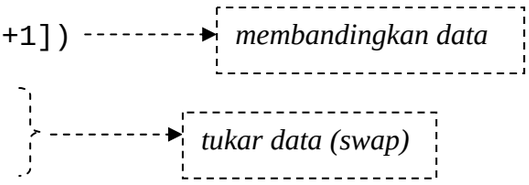
Data 1-2 : data[1] > data[2] 10 < 40
10 40 41 50 90
Data 2-3 : data[2] > data[3] 40 < 41
10 40 41 50 90

```
/* Program bubble sort ascending */
#include<stdio.h>
#include<conio.h>
main()
{
int i,n,temp,j;
int data[10];
clrscr();
/* ---memasukan data --- */
printf("Bubble sort ascending \n");
printf("Banyak data : ");
scanf("%i",&n);
for(i=1;i<=n;i++)
{
  printf("Data ke-%i : ",i);
  scanf("%i", &data[i]);
}
```

Output program :

```
Bubble sort ascending
Banyak data : 5
Data ke-1 :  40
Data ke-2 :  50
Data ke-3 :  10
Data ke-4 :  41
Data ke-5 :  90
Setelah pengurutan
10  40  41  50  90
```

```
/* --- proses pengulangan pembandingan & tukar data --- */
for(i=1;i<=(n-1);i++)
{
  for(j=1;j<=(n-i);j++)
  {
    if (data[j] > data[j+1])
    {
      temp=data[j];
      data[j]=data[j+1];
      data[j+1]=temp;
    }
  }
}
/* --- proses tampil data setelah urut --- */
printf("Setelah pengurutan \n");
for(i=1; i<=n;i++)
  printf("%i ",data[i]);
getch();
}
```




Untuk bubble sort data descending, program hanya diganti pada bagian

`if (data[j] > data[j+1])` menjadi `if (data[j] < data[j+1])`

## *PRAKTIKUM MINGGU 7*
## PENCARIAN / SEARCHING

- Pencarian elemen (searching) merupakan proses fundamental dalam pemrograman. Berbagai proses dalam aplikasi memerlukan searching, antara lain :
  - proses edit (perbaikan dan peremajaan (update) data) selalu didahului dengan pencarian akan posisi data yang akan diedit
  - proses insert (penyisipan data) memerlukan searching posisi dimana suatu data akan disisipkan
  - program pengolah kata (word atau text processing) menyediakan fasilitas Find, Replace, dan Goto yang pada hakekatnya memerlukan teknik pencarian.
- Teknik/metode pencarian :
  - Sequential search
  - Binary search

### 1. SEQUENTIAL SEARCH (PENCARIAN BERUNTUN)

Proses yang terjadi pada metode pencarian ini :

1. Membaca array data
2. Menentukan data yang dicari
3. Mulai dari data pertama sampai dengan data terakhir, data yang dicari dibandingkan dengan masing-masing data di dalam array
   a. Jika data yang dicari tidak ditemukan, maka semua data atau elemen array dibandingkan sampai selesai
   b. Jika data yang dicari ditemukan, maka perbandingan akan dihentikan

```
/* Program sequential search */
#include<stdio.h>
#include<conio.h>
main()
{
int i,n,posisi,cari,ketemu;
int data[10];
clrscr();
printf("Cari data sequential search \n");
printf("Banyak data : ");
scanf("%i",&n);
for(i=1;i<=n;i++)
{
  printf("Data ke-%i : ",i);
  scanf("%i", &data[i]);
}
printf("Data yang dicari : ");
scanf("%i", &cari);
ketemu = 0;
i=1;
while((ketemu==0) && (i<=n))
{ if(data[i]==cari)
   {
    ketemu=1;
    posisi=i;
   }
  else
  i=i+1; }
if (ketemu==1)
   printf("%i ditemukan pada posisi %i \n",cari,posisi);
else
   printf("%i tidak ditemukan \n",cari);
getch();
}
```

Output program :

```
Cari data sequential search
Banyak data : 5
Data ke-1 :  40
Data ke-2 :  50
Data ke-3 :  10
Data ke-4 :  41
Data ke-5 :  90
Data yang dicari : 10
10 ditemukan pada posisi 3
```

**2. BINARY SEARCH (PENCARIAN BINER)**

Hanya digunakan untuk pencarian data pada array yang sudah terurut. Prosesnya :

1. Tentukan posisi data awal dan posisi data akhir
2. Tentukan posisi data tengah dengan rumus = (posisi awal + posisi akhir)/2
3. Data yang dicari dibandingkan dengan data posisi tengah, jika sama berarti data ketemu. Proses berhenti.
4. Jika tidak sama, data yang dicari dibandingkan dengan data posisi tengah :
    a. jika lebih kecil dari data tengah proses dilakukan kembali dengan posisi akhir = posisi tengah – 1.
    b. jika lebih besar dari data tengah proses dilakukan kembali dengan posisi awal = posisi tengah + 1.

Contoh :

Terdapat 5 buah data yang nilainya sudah terurut ascending :
3 4 7 9 20
Mencari data 7

**Proses ke-1**

awal=1, akhir = 5, cari = 7
tengah = (1+5)/2=3
data[tengah] yaitu data[3] = 7
cari=data[3] ? → *ya, ketemu, proses selesai*

---

Contoh :

Terdapat 6 buah data yang nilainya sudah terurut ascending :
3 4 7 9 20 21
Mencari data 20

**Proses ke-1**

awal=1, akhir = 6, cari = 20
tengah = (1+6)/2=3
data[tengah] yaitu data[3] = 7
cari=data[3] ? → *tidak*
cari<data[3] ? *tidak*,karena 20 > 7 maka awal = tengah + 1 → awal=3+1=4 → *proses dilanjutkan*

**Proses ke-2**

awal=4, akhir = 6, cari = 20
tengah = (4+6)/2=5
data[tengah] yaitu data[5] = 20
cari=data[5] ? → ya, *ketemu, proses selesai*

---

```
/* Program binary search ascending */
#include<stdio.h>
#include<conio.h>
main()
{
int i,awal,akhir,tengah,dttengah,n,cari,posisi,ketemu;
int data[10];
clrscr();
printf("Cari data dengan binary search ascending \n");
printf("Banyak data : ");
scanf("%i",&n);
for(i=1;i<=n;i++)
{
  printf("Masukkan data ke-%i : ",i);
  scanf("%i", &data[i]);
}
printf("Data yang dicari : ");
scanf("%i", &cari);
```

```
awal=1;akhir=n;ketemu=0;
while ((ketemu==0) && (awal<=akhir))
{
  tengah=(awal+akhir)/2;
  dttengah=data[tengah];
  if (cari==dttengah)
  {
    ketemu=1;
    posisi=tengah;
  }
  else
  {
  if (cari<dttengah)
    akhir=tengah-1;
    else
    awal = tengah+1;
  }
 }
if (ketemu==0)
  printf("%i tidak ditemukan \n",cari);
else
  printf("%i ditemukan pada posisi %i\n",cari,posisi);
getch();
}
```

Output program :

```
Cari data binary search ascending
Banyak data : 6
Data ke-1 :  3
Data ke-2 :  4
Data ke-3 :  7
Data ke-4 :  9
Data ke-5 :  20
Data ke-6 :  21
Data yang dicari : 20
20 ditemukan pada posisi 5
```

Untuk binary search data descending, program hanya diganti pada bagian

`if (cari<dttengah)` menjadi `if (cari>dttengah)`

## Tambahan materi FUNGSI :

### REKURSIF

- Rekursif adalah suatu fungsi yang dapat memanggil dirinya sendiri, artinya fungsi tersebut dipanggil di dalam tubuh fungsi itu sendiri.

contoh :
ketentuan faktorial secara rekursif :
n = 0 maka n! = 1
n > 0 maka n! = n * (n – 1)!
misal 4! = 4 * 3!

```
/* ------ Program faktorial secara rekursif ------ */
#include<stdio.h>
#include<conio.h>
int fak(int x);  /*deklarasi fungsi*/
main()
{
int n,m;
clrscr();
printf("Masukan bilangan : ");
scanf("%i",&n);
m=fak(n);
printf("Faktorial dari %i adalah %i ",n,m);
getch();
}

int fak(int x)  /* definisi fungsi */
{
  if(x==0) return(1);
    else return (x*fak(x-1));
}
```

Output program :

```
Masukkan bilangan : 4
Faktorial dari 4 adalah 24
```

```
Algoritma faktorialrekursif
Deklarasi
    n,m : integer
    function fak(input x : integer) → integer
Deskripsi
    read(n)
    m=fak(n)
    write(m)

function fak(input x : integer) → integer
Deklarasi
    if x=0  then
        return(1)
    else
         return(x * fak(x – 1));
    endif
```

PENCARIAN, PENGURUTAN

```
#include<stdio.h>
#include<conio.h>
main()
{
  char nama[10] [25];
  int nim[10],temp;
  int nilai[10];
  int cari;
  int i,n,j,l,k;

  clrscr();
  printf("Banyak data :");
  scanf("%i",&n);
  printf("\n");
  for(i=0;i<n;i++)
    {
    printf("Data ke-%i \n",i);
    printf("Nama  : ");scanf("%s",&nama[i]);
    printf("Nim   : ");scanf("%i",&nim[i]);
    printf("Nilai : ");scanf("%i",&nilai[i]);
    printf("\n");
    }
  printf("\n");
  clrscr();
  gotoxy(10,2);printf("----------------------------");
  gotoxy(10,3);printf("NO   NIM   NAMA     NILAI \n");
  gotoxy(10,4);printf("----------------------------");
  for(i=0;i<n;i++)
    {
    gotoxy(10,5+i);printf("%2i.",i);
    gotoxy(15,5+i);printf("%i  ",nim[i]);
    gotoxy(20,5+i);printf("%s  ",nama[i]);
    gotoxy(30,5+i);printf("%3i  ",nilai[i]);
    }
  gotoxy(10,6+i);printf("----------------------------");
  /* cari data berdasarkan*/
  printf("\n");
  printf("cari nilai  : ");scanf("%i",&cari);
  for (i=0;i<n;i++)
  {
    if (nilai[i]==cari)
      {
      printf("%i  ",nim[i]);
      printf("%s  ",nama[i]);
      printf("%3i  ",nilai[i]);}
      printf("\n");
  }
     /* urut data berdasarkan nama*/
  printf("\n");
  getch();
  for (i=1;i<=n;i++)
  {
    for (j=i;j<n;j++)
    {
      for(l=0;l<20;l++)
      {
        if(nama[i][l] < nama[j+1][l])
          {              break;}
        if(nama[i][l] > nama[j+1][l])
          {
            for(k=0;k<20;k++)
            {
            temp=nama[i][k];
            nama[i][k]=nama[j+1][k];
            nama[j+1][k]=temp;
            }
            break;
          }
       }
      }
```

```
}
  printf("Setelah urut \n");
  for(i=0;i<n;i++)
    printf("%s \n",nama[i]);
     /* urut data berdasarkan nim*/
  printf("\n");

    for (i=0;i<n-1;i++)
     {
    for (j=0;j<n-i;j++)
      {
      if (nim[j] > nim[j+1])
          {
           temp=nim[j];
           nim[j]=nim[j+1];
           nim[j+1]=temp;
           }
      }
      }

  printf("Setelah urut \n");
  for(i=0;i<n;i++)
   {
    printf("%i ",nim[i]);
    printf("%s ",nama[i]);
    printf("%3i ",nilai[i]);
    printf("\n");
   }

  getch();
}
```