MODUL PRAKTIKUM
ALGORITMA & PEMROGRAMAN

Disusun Oleh:
Achmad Fiqhi Ibadillah S.T.,M.Sc.

PROGRAM STUDI TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS TRUNOJOYO MADURA
2019

PRAKTIKUM MINGGU I

PEMILIHAN KONDISI

I. Tujuan Praktikum:

1. Mahasiswa dapat menjelaskan tentang operator kondisi
2. Dapat menerapkan pernyataan if-else
3. Dapat menerapkan pernyataan switch-case

II. Dasar Teori

A. OPERATOR KONDISI

Operator yang digunakan untuk menghasilkan kondisi benar dan salah, bisa berupa operator relasi dan bisa juga berupa operator logika. Berikut ini dibahas masing-masing jenis operator serta tabel prioritas masing-masing operator.

1. OPERATOR RELASI

Operator relasi biasa dipakai untuk membandingkan dua buah nilai. Hasil pembandingan berupa keadaan benar atau salah. Keseluruhan operator relasi pada C ditunjukkan pada Tabel 1-1.

Tabel 1-1. Operasi relasi

Operator	Makna
>	Lebih dari
>=	Lebih dari atau sama
<	dengan Kurang dari
<=	Kurang dari atau sama
==	dengan Sama dengan
!=	Tidak sama dengan

2. OPERATOR LOGIKA

Operator logika biasa dipakai untuk menghubungkan ekspresi relasi. Keseluruhan operator logika ditunjukkan pada tabel 1-2.

Tabel 1-2. Operator logika

Operator	Makna
&&	dan (AND)
||	atau (OR)
!	tidak (NOT)

3. PRIORITAS OPERATOR LOGIKA DAN RELASI

Tabel berikut ini memberikan penjelasan singkat mengenai prioritas di antara berbagai operator logika dan operator relasi.

Tabel 1-3 Prioritas operator logika dan relasi

Tertinggi	!	>	<	<
	>	>=		
	==			
	!=			
Terendah				

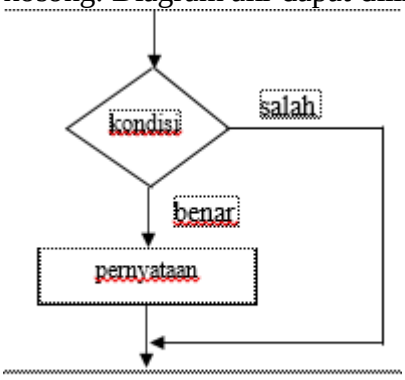
B. PERNYATAAN KONDISI

1. Pernyataan IF

Pernyataan if mempunyai bentuk umum :

if (kondisi)
pernyataan;

Bentuk ini menyatakan :
Jika kondisi yang diseleksi adalah benar (bernilai logika = 1), maka pernyataan yang mengikutinya akan diproses.
Sebaliknya, jika kondisi yang diseleksi adalah tidak benar (bernilai logika = 0), maka pernyataan yang mengikutinya tidak akan diproses.
Mengenai kodisi harus ditulis diantara tanda kurung, sedangkan pernyataan dapat berupa sebuah pernyataan tunggal, pernyataan majemuk atau pernyataan kosong. Diagram alir dapat dilihat seperti gambar 3.1



Gambar 3.1. Diagram alur if

Contoh penggunaan pernyataan if adalah untuk menentukan besarnya potongan harga yang diterima oleh seorang pembeli, berdasarkan kriteria :
tidak ada potongan harga jika total pembelian kurang dari Rp. 100.000 (dalam hal ini potongan harga diinisialisasi dengan nol).
bila total pembelian lebih dari atau sama dengan Rp. 100.000, potongan harga yang diterima dirubah menjadi sebesar 5% dari total pembelian.

CONTOH PERMASALAHAN 1:

Contoh penggunaan if untuk menghitung nilai discount

```
/* File program : discount.c
Contoh penggunaan if untuk menghitung nilai discount */
#include <stdio.h>
main()
{
    double total_pembelian, discount = 0;
    /* discount diinisialisasi dengan nilai 0 */

    printf("Total pembelian    = Rp ");
    scanf("%lf", &total_pembelian);

    if(total_pembelian >= 100.000) discount
       = 0.05 * total_pembelian;
    printf("Besarnya discount = Rp %.2lf\n", discount);
}
```

Untuk pernyataan if yang diikuti dengan pernyataan majemuk, bentuknya adalah sebagai berikut :

if (kondisi)
{ /* tanda awal pernyataan majemuk*/
pernyataan-1;
pernyataan–2;
.
.
.
pernyataan-n;
} /* tanda akhir pernyataan majemuk */

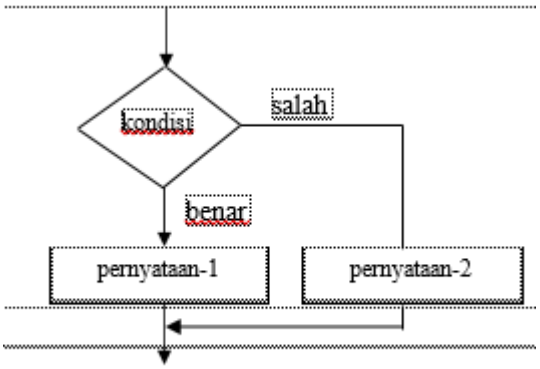
Pernyataan-pernyataan yang berada dalam tanda kurung { dan } akan dijalankan hanya bila kondisi if bernilai benar.

2. **Pernyataan IF-ELSE**

Pernyataan if-else memiliki bentuk :

if (kondisi) pernyataan-1;
else
pernyataan-2;

Diagram alir dapat dilihat seperti gambar 3.2.



Gambar 3.2. Diagram alir if-else

Arti dari pernyataan if-else :

- Jika kondisi benar, maka pernyataan-1 dijalankan. Sedangkan bila kondisi bernilai salah, maka pernyataaan-2 yang dijalankan.
- Masing-masing pernyataan-1 dan pernyataan-2 dapat berupa sebuah pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong.
- Contoh penggunaan pernyataan if-else adalah untuk menyeleksi nilai suatu bilangan pembagi. Jika nilai bilangan pembagi adalah nol, maka hasil pembagian dengan nilai nol akan mendapatkan hasil tak berhingga. Jika ditemui nilai pembaginya nol, maka proses pembagian tidak akan dilakukan.

CONTOH PERMASALAHAN 2

```
/* File program : bagi.c
 Pemakaian if-else untuk menyeleksi bilangan pembagi */
 #include <stdio.h>
 main()
 {
    float a, b;

    printf("Masukkan nilai a : ");
    scanf("%f", &a);
    printf("Masukkan nilai b : ");
    scanf("%f", &b);

    if (b == 0)
       printf("\n%g dibagi dengan nol = TAK BERHINGGA\n", a);
    else
       printf("\n%g dibagi dengan %g  = %g\n", a, b, a/b);
 }
```

3. **PERNYATAAN IF DALAM IF**

Di dalam suatu pernyataan if (atau if-else) bisa saja terdapat pernyataan if (atau if- else) yang lain. Bentuk seperti ini dinamakan sebagai nested if. Secara umum, bentuk dari pernyataan ini adalah sebagai berikut :

```
if (kondisi-1)
        if (kondisi-2)
        .
                if (kondisi-n)
                        pernyataan;
                else
                        pernyataan;
                .
        else
                pernyataan;
else
        pernyataan;
```

- Kondisi yang akan diseleksi pertama kali adalah kondisi yang terluar (kondisi-1). Jika kondisi-1 bernilai salah, maka statemen else yang terluar (pasangan if yang bersangkutan) yang akan diproses. Jika else (pasangannya tsb) tidak ditulis, maka penyeleksian kondisi akan dihentikan.

- Jika kondisi-1 bernilai benar, maka kondisi berikutnya yang lebih dalam (kondisi-2) akan diseleksi. Jika kondisi-2 bernilai salah, maka statemen else pasangan dari if yang bersangkutan yang akan diproses. Jika else (untuk kondisi-2) tidak ditulis, maka penyeleksian kondisi akan dihentikan.
- Dengan cara yang sama, penyeleksian kondisi akan dilakukan sampai dengan kondisi- n, jika kondisi-kondisi sebelumnya bernilai benar.

CONTOH PERMASALAHAN 3

```
/* File program : diskriminan1.c
Program untuk menghitung diskriminan dan akar-akar persamaan
kuadrat menggunakan if bersarang */

#include <stdio.h>
#include <math.h>

main()
{
    float a, b, c, d = 0;
    double x1, x2, imaginair;

    printf("MENCARI AKAR-AKAR PERSAMAAN KUADRAT a+bx+c=0\n");
    printf("\nMasukkan nilai a : ");
    scanf("%f", &a); printf("Masukkan
    nilai b : "); scanf("%f", &b);
    printf("Masukkan nilai c : ");
    scanf("%f", &c);
    d = b*b-4*a*c;  /* menghitung diskriminan */
    if (d >= 0)
       if (d == 0)
       {
          x1 = -b / (2 * a);
          printf("\nDua akar real kembar yaitu : \n");
          printf("x1 = x2 = %g\n", x1);
       }
       else
       {
          x1 = (-b + sqrt(d))/(2*a);
          x2 = (-b - sqrt(d))/(2*a);
          printf("\nDua akar real berlainan yaitu :\n");
          printf("x1 = %g\n", x1);
          printf("x2 = %g\n", x2);

       }
       else
       {
          imaginair = (sqrt(-d)/(2*a)); x1 = -b/(2*a);
          printf("\nDua akar imaginair berlainan yaitu : \n");
          printf("x1 = %g + %gi\n", x1, imaginair);
          printf("x2 = %g - %gi\n", x1, imaginair);
       }
    }
```

4. **PERNYATAAN ELSE-IF**

   Contoh implementasi nested if ini misalnya pembuatan sebuah program kalkulator sederhana. User memberikan masukan dengan format : operand1 operator operand2

Jenis operasi yang dikenakan bergantung pada jenis operator yang dimasukkan oleh user. Oleh karena itu program akan mengecek apakah operator berupa tanda '*', '/', '+', ataukah tanda '-' .

- Jika operator berupa tanda '*' maka operand1 akan dikalikan dengan operand2.
- Jika operator berupa tanda '/' maka operand1 akan dibagi dengan operand2.
- Jika operator berupa tanda '+' maka operand1 akan dijumlahkan dengan operand2.
- Jika operator berupa tanda '-' maka operand1 akan dikurangi dengan operand2.
- Kalau operator yang dimasukkan bukan merupakan salah satu dari jenis operator di atas, maka ekspresi tersebut tidak akan diproses, dan user akan mendapatkan pesan berupa : "Invalid operator !"

CONTOH PERMASALAHAN 4

```
/* File program : kalkulator1.c
Contoh penggunaan else if untuk mengimplementasikan program
kalkulator sederhana */
#include <stdio.h>
main()
{
    int  valid operator = 1;
    /* valid_operator diinisialisasi dengan logika 1 */
    char  operator;
    float  number1, number2, result;
    printf("Masukkan 2 buah bilangan dan sebuah operator\n");
    printf("dengan format : number1 operator number2\n\n");
    scanf("%f %c %f", &number1, &operator, &number2);

    if(operator == '*')
        result = number1 * number2;
    else if(operator == '/')
        result = number1 / number2;
    else if(operator == '+')
        result = number1 + number2;
    else if(operator == '-')
        result = number1 - number2;
    else
        valid operator = 0;

    if(valid operator)
        printf("\n%g %c %g is %g\n", number1, operator,
                number2, result );
    else
        printf("Invalid operator!\n");
}
```

5. KONDISI SWITCH-CASE

Pernyataan switch-case merupakan pernyataan yang dirancang khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah alternatif, misalnya untuk menggantikan pernyataan if bertingkat.

Bentuk umum pernyataan switch adalah :

```
switch (ekspresi)
{
        case konstanta-1:
                pernyataan-1;
                ......
                break;
        case konstanta-2:
                .
                .
                .
        case konstanta-n:
                pernyataan-n;
                .......
                break;
        default:
                .......
                .......
                break;
}
```

Dengan ekspresi dapat berupa ekspresi bertipe integer atau bertipe karakter. Demikian juga konstanta-1, konstanta-2, …, konstanta-n dapat berupa konstanta integer atau karakter. Setiap pernyataan-i (pernyataan-1, … , pernyataan-n) dapat berupa pernyataan tunggal ataupun pernyataan jamak. Dalam hal ini urutan penulisan pernyataan case tidak berpengaruh. Proses penyeleksian berlangsung sebagai berikut :

- pengujian pada switch akan dimulai dari konstanta-1. Kalau nilai konstanta-1 cocok dengan ekspresi maka pernyataan-1 dijalankan. Kata kunci break harus disertakan di bagian akhir setiap pernyataan case, yang akan mengarahkan eksekusi ke akhir switch.
- Kalau ternyata pernyataan-1 tidak sama dengan nilai ekspresi, pengujian dilanjutkan pada konstanta-2, dan berikutnya serupa dengan pengujian pada konstanta-1.
- Jika sampai pada pengujian case yang terakhir ternyata tidak ada kecocokan, maka pernyataan yang mengikuti kata kunci default yang akan dieksekusi. Kata kunci default ini bersifat opsional.
- Tanda kurung kurawal tutup (}) menandakan akhir dari proses penyeleksian kondisi case.

CONTOH PERMASALAHAN 5

Di bawah ini contoh program pemakaian pernyataan switch untuk menggantikan if-else bertingkat pada program kalkulator1.c di atas.

```
/* File program : kalkulator2.c
Contoh penggunaan pernyataan switch untuk mengimplementasikan
kalkulator sederhana */
#include <stdio.h>
main()
{
    int  valid_operator = 1;
    char  operator;
    float  number1, number2, result;

    printf("Masukkan 2 buah bilangan dan sebuah operator\n");
    printf("dengan format : number1 operator number2\n\n");
    scanf("%f %c %f", &number1, &operator, &number2);
    switch(operator) {
    case '*' : result = number1 * number2; break;
    case '/' : result = number1 / number2; break;
    case '+' : result = number1 + number2; break;
    case '-' : result = number1 - number2; break;
    default : valid_operator = 0;
    }
    if(valid_operator)
        printf("%g %c %g is %g\n", number1, operator,
number2,result);
    else
        printf("Invalid operator!\n");
}
```

PRAKTIKUM MINGGU II

PENGULANGAN

I. Tujuan Praktikum:

1. Mahasiswa dapat menjelaskan tentang pengulangan dalam bahasa C
2. Mahasiswa dapat menerapkan pengulangan for
3. Mahasiswa dapat menerapkan pengulangan while
4. Mahasiswa dapat menerapkan pengulangan while-do

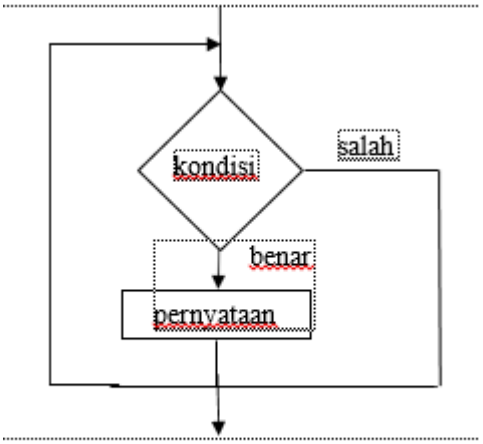
II. Dasar Teori

A. PERNYATAAN WHILE

Pada pernyataan while, pengecekan terhadap loop dilakukan di bagian awal (sebelum tubuh loop). Lebih jelasnya, bentuk pernyataan while adalah sebagai berikut:

```
while (kondisi)
   pernyataan;
```

dengan pernyataan dapat berupa pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong. Proses pengulangan terhadap pernyataan dijelaskan pada gambar berikut :



Gambar 2.1. Diagram alir while

Dengan melihat gambar 2.1, tampak bahwa ada kemungkinan pernyataan yang merupakan tubuh loop tidak dijalankan sama sekali, yaitu kalau hasil pengujian kondisi while yang pertama kali ternyata bernilai salah.

Contoh pemakaian while misalnya untuk mengatur agar tombol yang ditekan oleh pemakai program berupa salah satu diantara 'Y','y', 'T' atau 't'. Impelementasinya:

CONTOH PERMASALAHAN 6

```
/*File program :  pilihan.c Untuk
membaca tombol Y atau T */
#include <stdio.h>
main()
{
      char pilihan;
      /* diberi nilai salah lebih dahulu */
      int sudah benar = 0;

      printf("Pilihlah Y atau T.\n");
 /* program dilanjutkan jika tombol Y,y,T atau t ditekan */
      while(!sudah benar)
      {
         pilihan = getchar();        /* baca tombol */
         sudah benar = (pilihan == 'Y') || (pilihan == 'y')||
            (pilihan == 'T') || (pilihan == 't');
      }

      /* memberi keterangan tentang pilihan */
      switch(pilihan)
      {
      case 'Y':
      case 'y':
            puts("\nPilihan anda adalah Y");
            break;
      case 'T':
      case 't':
            puts("\nPilihan anda adalah T");
      }
 }
```

Inisialisasi terhadap variabel sudah_benar yang akan dijalankan pada kondisi while dengan memberi nilai awal bernilai false (sudah_benar = 0) dimaksudkan agar tubuh loop

pilihan = getchar( ); /* baca tombol */
sudah_benar = (pilihan == 'Y') || (pilihan== 'y')|| (pilihan == 'T') || (pilihan == 't'); }

dijalankan minimal sekali.

Contoh lain pemakaian while dapat dilihat pada program yang digunakan untuk menghitung banyaknya karakter dari kalimat yang dimasukkan melalui keyboard (termasuk karakter spasi). Untuk mengakhiri pemasukan kalimat, tombol ENTER (‘\n’) harus ditekan. Karena itu, tombol ENTER inilah yang dijadikan kondisi penghitungan jumlah spasi maupun karakter seluruhnya. Lengkapnya, kondisi yang dipakai dalam while berupa :

while((kar = getchar()) != '\n')

Ungkapan di atas mempunyai arti :

- Bacalah sebuah karakter dan berikan ke variabel kar
- Kemudian bandingkan apakah karakter tersebut = '\n' (ENTER)

Ungkapan menghasilkan nilai benar jika tombol yang ditekan bukan ENTER.Pada program kalau tombol yang ditekan bukan ENTER , maka :

- Jumlah karakter dinaikkan sebesar satu melalui pernyataan : jumkar++;
- Kalau karakter berupa SPASI, maka jumlah spasi dinaikkan sebesar satu, melalui pernyataan : if (kar == ' ') jumspasi++;

CONTOH PERMASALAHAN 7

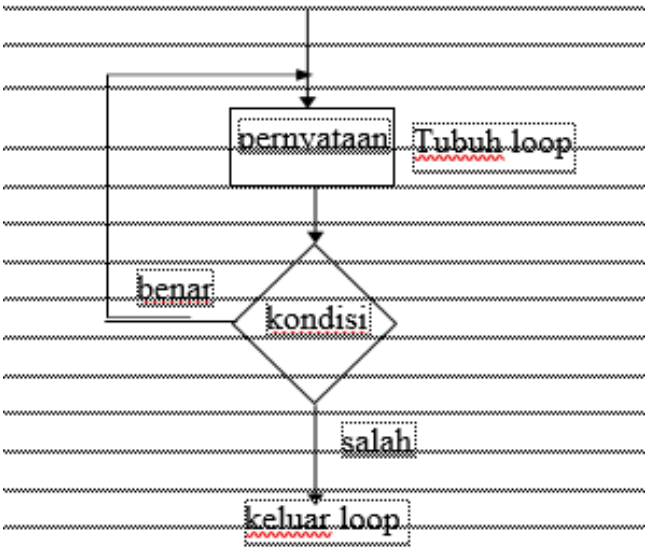
```
/* File program : jumkar.c
Menghitung jumlah kata dan karakter dalam suatu kalimat
*/ #include <stdio.h>
main()
{
char kar;
int jumkar = 0, jumspasi = 0;

puts("Masukkan sebuah kalimat dan akhiri dgn ENTER.\n");
puts("Saya akan menghitung jumlah karakter ");
puts("padakalimat tersebut.\n");

while((kar = getchar()) != '\n')
{

jumkar++;
if (kar == ' ') jumspasi++;
}

printf("\nJumlah karakter              = %d", jumkar);
printf("\nJumlah SPASI                 = %d\n\n", jumspasi);
}
```

B. PERNYATAAN DO-WHILE

Bentuk pernyataan do-while:

```
do
   pernyataan;
while (kondisi)
```

Pada pernyataan do-while, tubuh loop berupa pernyataan,dengan pernyataan bisa berupa pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong. Pada pernyataan do, mula-mula pernyataan dijalankan. Selanjutnya, kondisi diuji. Sendainya kondisi bernilai benar, maka pernyataan dijalankan lagi, kemudian kondisi diperiksa kembali, dan seterusnya. Kalau kondisi bernilai salah pada saat dites, maka pernyataan tidak dijalankan lagi. Untuk lebih jelasnya dapat dilihat pada Gambar 2.2. Berdasarkan Gambar 2.2 terlihat bahwa tubuh loop minimal akan dijalankan sekali.



Gambar 2.2. Diagram alir do-while

Program berikut memberikan contoh pemakaian do-while untuk mengatur penampilan tulisan "BAHASA C" sebanyak sepuluh kali.
Contoh:

```
i = 0;
do
{
        puts("BAHASA C");
         i++;
} while(i<10);
```

Pada program di atas, variabel pencacah dipakai untuk menghitung jumlah tulisan yang sudah ditampilkan pada layar. Selama nilai pencacah kurang dari 10, maka perintah

```
puts("BAHASA C");
```

akan dilaksanakan kembali.

Penanganan pembacaan tombol pada contoh program pilihan.c yang memakai while di atas, kalau diimplementasikan dengan memakai do-while adalah sebagai berikut:

CONTOH PERMASALAHAN 8

```
/* File program : pilihan2.c Untuk membaca tombol Y atau T */

#include <stdio.h>
main()
{
      char pilihan; int
      sudah benar;

      printf("Pilihlah Y atau T.\n");

/* program dilanjutkan kalau tombol Y,y,T atau t ditekan */
      do
      {
          pilihan = getchar( ); /* baca tombol */
          sudah_benar = (pilihan == 'Y') || (pilihan== 'y')||
           (pilihan == 'T') || (pilihan == 't');
      } while(! sudah benar);

      /* memberi keterangan tentang pilihan */
      switch(pilihan)
      {
      case 'Y':
      case 'y':
            puts("\nPilihan anda adalah Y");
            break;
      case 'T':
      case 't':
            puts("\nPilihan anda adalah T");
      }
}
```

Mula-mula tombol dibaca dengan menggunakan getchar() dan kemudian diberikan ke variabel pilihan. Sesudah itu, variabel sudah_benar akan diisi dengan nilai benar (1) atau salah (0) tergantung dari nilai pilihan. Kalau pilihan berisi salah satu diantara ‘Y’,‘y’, ‘T’ atau ‘t’, maka sudah berisi salah satu diantara ‘Y’, ‘y’, ‘T’ atau ‘t’, maka sudah_benar akan berisi benar. Nilai pada vaiabel sudah_benar ini selanjutnya dijadikan sebagai kondisi do-while. Pengulangan terhadap pembacaan tombol akan dilakukan kembali selama sudah_benar benilai salah.

C. PERNYATAAN FOR

Mengulang suatu proses merupakan tindakan yang banyak dijumpai dalam pemrograman. Pada semua bahasa pemrograman, pengulangan proses ditangani dengan suatu mekanisme yang disebut loop. Dengan menggunakan loop, suatu proses yang berulang misalnya menampilkan tulisan yang sama seratus kali pada layar dapat diimpelementasikan dengan kode program yang pendek.

Pernyataan pertama yang digunakan untuk keperluan pengulangan proses adalah pernyataan for. Bentuk pernyataan ini :

```
for (ungkapan1; ungkapan2; ungkapan3)
                pernyataan;
```

Kegunaan dari masing-masing ungkapan pada pernyataan for.

- Ungkapan1 : digunakan untuk memberikan inisialisasi terhadap variabel pengendali loop.
- Ungkapan2 : dipakai sebagai kondisi untuk keluar dari loop.
- Ungkapan3 : dipakai sebagai pengatur kenaikan nilai variabel pengendali loop.

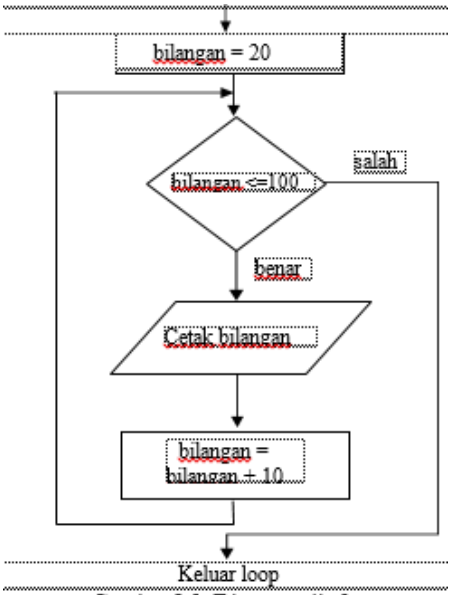
Ketiga ungkapan dalam for tersebut harus dipisahkan dengan tanda titik koma (;). Dalam hal ini pernyatan bisa berupa pernyataan tunggal maupun jamak. Jika pernyataannya berbentuk

```
for (ungkapan1; ungkapan2; ungkapan3)
{
   pernyataan;
   pernyataan;
   .
   .
   .
}
```

jamak, maka pernyataan-pernyataan tersebut harus diletakkan di antara kurung kurawal buka ({) dan kurung kurawal tutup (}), sehingga formatnya menjadi:
Contoh penggunaan for, misalnya untuk menampilkan deretan angka sebagai berikut :

20
30
40
50
.
100

Untuk keperluan ini, pernyataan for yang digunakan berupa :
for (bilangan = 20; bilangan <= 100; bilangan += 10) printf("%d\n", bilangan);
Kalau digambarkan dalam bentuk diagram alir, akan terlihat sbb:



Gambar 2.3. Diagram alir for

CONTOH PERMASALAHAN 9

```
/* File program : for1.c
Contoh pemakaian for untuk membentuk deret naik */
#include <stdio.h>
main()
{
        int bilangan;

        for(bilangan = 20; bilangan <= 100; bilangan += 10)
              printf("%d\n", bilangan);
}
```

Pada program di atas, kenaikan terhadap variabel pengendali loop sebesar 10 (positif), yang dinyatakan dengan ungkapan

bilangan += 10

yang sama artinya dengan

bilangan = bilangan + 10

Pada contoh yang melibatkan pernyataan for di atas, kenaikan variabel pengendali loop berupa nilai positif. Sebenarnya kenaikan terhadap variabel pengendali loop bisa diatur bernilai negatif. Cara ini dapat digunakan untuk memperoleh deret sebagai berikut:

60
50
40
30
20
10

Untuk itu selengkapnya program yang dibutuhkan adalah sebagai berikut:

CONTOH PERMASALAHAN 10

```
/* File program : for2.c
   Contoh pemakaian for untuk membentuk deret turun */
#include <stdio.h>
main()
{
      int bilangan;

      for (bilangan = 60; bilangan >= 10; bilangan -= 10)
            printf("%d\n", bilangan);
}
```

Kadang-kadang dijumpai adanya pernyataan for yang tidak mengandung bagian ungkapan yang lengkap (beberapa ungkapan dikosongkan). Dengan cara ini, pernyataan

for (bilangan = 20; bilangan <= 100; bilangan += 10) printf("%d\n", bilangan);

dapat ditulis menjadi :

```
bilangan = 20;          /* inisialisasi di luar for */
for (bilangan <= 100)
{
        printf("%d\n", bilangan); bilangan += 10;
}
```

Tampak bahwa ungkapan yang biasa dipakai untuk inisialisasi variabel pengendali loop tak ada. Sebagai gantinya pengendalian loop diatur sebelum pernyataan for, berupa

bilangan = 20;

Pengosongan ini juga dilakukan pada ungkapan yang biasa dipakai untuk menaikkan nilai variabel pengendali loop. Sebagai gantinya, di dalam tubuh loop diberikan pernyataan untuk menaikkan nilai variabel pengendali loop, yaitu berupa

bilangan += 10;

Ungkapan yang tidak dihilangkan berupa bilangan <=100. Ungkapan ini tetap disertakan karena dipakai sebagai kondisi untuk keluar dari loop. Sesungguhnya ungkapan yang dipakai sebagai kondisi keluar dari loop juga bisa dihilangkan, sehingga bentuknya menjadi

```
for (;;)
        pernyataan
```

Suatu pertanyaan mungkin timbul "Lalu bagaimana caranya kalau ingin keluar dari loop pada bentuk di atas?". Caranya adalah dengan menggunakan pernyataan yang dirancang khusus untuk keluar dari loop. Mengenai hal ini akan dibahas pada sub bab yang lain.

1. FOR BERSARANG

   Dalam suatu loop bisa terkandung loop yang lain. Loop yang terletak di dalam loop biasa disebut dengan loop di dalam loop (nested loop). Salah satu contoh nested loop misalnya pada permasalahan untuk membuat tabel perkalian:

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	4	6	8	10	12	14	16
3	3	6	9	12	15	18	21	24
4	4	8	12	16	20	24	28	32
5	5	10	15	20	25	30	35	40
6	6	12	18	24	30	36	42	48
7	7	14	21	28	35	42	49	56
8	8	16	24	32	40	48	56	64

CONTOH PERMASALAHAN 11

```
/* File program : tblkali.c
Loop for bersarang untuk membuat tabel perkalian */
#include <stdio.h>
#define MAKS 8

main()
{
     int baris, kolom, hasil kali;

     for (baris = 1; baris <= MAKS; baris++)
     {
          for (kolom = 1; kolom <= MAKS; kolom++)
          {
               hasil kali = baris * kolom;
               printf ("%2d", hasil kali);
          }
          printf("\n");  /* pindah baris */
     }
}
```

Bagian yang terletak dalam bingkai di depan dapat dapat diperoleh melalui

for (baris = 1; baris <= MAKS; baris++)
{
hasil_kali = baris * kolom; printf ("%2d", hasil_kali);
}

Dengan MAKS didefinisikan bernilai 8. Bagian loop yang terdalam :

for (kolom = 1; kolom <= MAKS; kolom++)
{
hasil_kali = baris * kolom; printf ("%2d", hasil_kali);
}

Digunakan untuk mencetak suatu deret hasil perkalian dalam satu baris.Untuk berpindah ke baris berikutnya, pernyataan yang digunakan yaitu:

printf("\n");

Adapun pencetakan untuk semua baris dikendalikan melalui

for (baris = 1; baris <= MAKS; baris++)

Pernyataan di atas mempunyai arti “dari baris ke-1 sampai dengan baris ke-MAKS”.

2. PERNYATAAN BREAK

Pernyataan break sesungguhnya telah diperkenalkan pada pernyataan switch. Pernyataan ini berfungsi untuk keluar dari loop for, do-while dan while. Sedangkan pada switch yaitu untuk menuju ke akhir (keluar dari) struktur switch. Sebagai contoh dapat dilihat pada gambar 4.4. Kalau pernyataan break dijalankan maka eksekusi akan dilanjutkan ke pernyataan yang terletak sesudah akhir tubuh loop for.

```
for ( ; ; )
{
        .
        .
  if ( …… )
    break;
  }                    /* akhir tubuh loop for */

puts(“\nSelesai…”);
```

Pada contoh potongan program berikut, pembacaan dan penampilan terhadap tombol yang ditekan akan berakhir kalau tombol yang ditekan adalah ENTER (‘\n’). Pernyataan yang digunakan untuk keperluan ini :

if (kar == ‘\n')

break; /* keluar dari loop for */

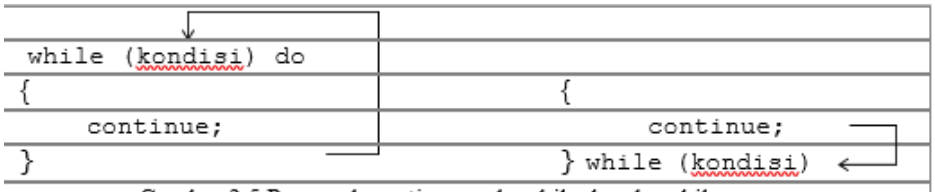
Yang menyatakan “Jika tombol yang ditekan berupa ENTER, maka keluarlah dari loop for”. Untuk lebih jelasnya, perhatikan program di bawah ini:

CONTOH PERMASALAHAN 12

```
/* File program : tamat.c
Pemakaian break untuk keluar dari looping */
#include <stdio.h>
main()
{
     char kar;

     printf("Ketik sembarang kalimat");
     printf(" dan akhiri dengan ENTER\n\n");
     for ( ; ; )
     {
       kar = getchar();
       if(kar == '\n')
           break;
     }
     printf("Selesai\n");
}
```

3. PERNYATAAN CONTINUE

Pernyataan continue digunakan untuk mengarahkan eksekusi ke iterasi (proses) berikutnya pada loop yang sama. Pada do-while dan while, pernyataan continue menyebabkan eksekusi menuju ke kondisi pengujian pengulangan, seperti yang dilukiskan pada Gambar 4.5. Pada loop for, pernyataan contunue menyebabkan bagian penaik variabel pengendali loop dikerjakan (ungkapan3 pada struktur for) dan kondisi untuk keluar dari loop for (ungkapan2 pada struktur for) diuji kembali.
Program ini digunakan untuk memasukkan data harus diulangi dan hal ini dikendalikan dengan continue. Untuk mengakhiri pemasukan data, data yang dimasukkan harus bernilai kurang dari 0. Perlu diketahui kondisi bernilai 1.



Gambar 2.5 Pengaruh continue pada while dan do-while

Menyatakan bahwa kondisi selalu dianggap benar. Untuk keluar dari loop, pernyataan yang digunakan berupa break.
Pengaruh continue pada loop for diperlihatkan pada dibawah ini. Program ini dipakai untuk menampilkan bilangan ganjil yang terletak antara 7 sampai dengan 25, kecuali 15.

CONTOH PERMASALAHAN 13

```
/* File program : ganjil.c
menampilkan bilangan ganjil antara 7 - 25 kecuali 15 */
#include <stdio.h>
main()
{
     int x;

     for (x = 7; x <= 25; x += 2)
     {
       if  (x == 15)
          continue;
       printf("%4d", x);
     }
     printf("\n");
}
```

Pada program di atas, untuk menghindari agar nilai 15 tidak ditampilkan ke layar, pernyataan yang digunakan berupa

if (x == 15)
continue;

Artinya, jika kondisi x == 15 bernilai benar, pernyataan continue menyebabkan pernyataan sisanya yaitu

printf("%d",x);

diabaikan dan eksekusi diarahkan kepada ungkapan:

x += 2

dan kemudian menguji kondisi:

x <= 25

Pada program di atas, pernyataan:

```
for (x = 7; x <= 25; x += 2)
{
if(x == 15) continue;
printf("%4d", x);
}
```

dapat ditulis dalam bentuk lain sebagai berikut:

```
for (x = 7; x <= 25; x += 2) if (x != 15)
        printf("%4d", x);
```

4. PERNYATAAN GOTO

Pernyataan goto merupakan intruksi untuk mengarahkan eksekusi ke pernyataan yang diawali dengan suatu label. Label sendiri berupa suatu pengenal (identifier) yang diikuti dengan tanda titik dua (:)

Contoh pemakaian goto ditujukan pada program dibawah ini:

```
Pernyataan
goto cetak;
```

Mengisyaratkan agar eksekusi dilanjutkan ke pernyataan yang diawali dengan label

cetak:

Pernyataan

```
if (++pencacah <= 10) goto cetak;
```

Mempunyai arti :

- Naikkan nilai pencacah sebesar 1
- Kemudian, jika pencacah kurang dari atau sama dengan 10 maka eksekusi menuju ke label cetak.

Penerapan goto biasanya dilakukan pada loop di dalam loop (nested loop), dengan tujuan memudahkan untuk keluar dari loop terdalam menuju ke pernyataan yang terletak di luar loop terluar.

PRAKTIKUM MINGGU III

ARRAY DAN STRING

I. Tujuan Praktikum:

1. Mahasiswa mengerti tentang array dimensi satu dan dua
2. Mahasiswa dapat menerapkan array dimensi satu dan dua
3. Mahasiswa mengerti tentang variabel string dan I/O string
4. Mahasiswa dapat menerapkan variabel string dan I/O string

II. Dasar Teori

A. ARRAY

1. Array Dimensi satu

Suatu array berdimensi satu dideklarasikan dalam bentuk umum berupa :

tipe_data nama_var[ukuran];

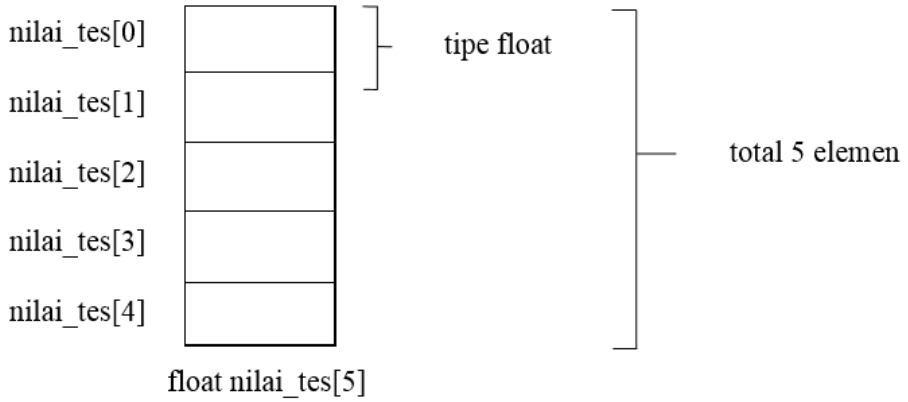
dengan :

- *tipe_data* : untuk menyatakan tipe dari elemen array, misalnya int, char, float.
- *nama_var* : nama variabel array
- *ukuran* : untuk menyatakan jumlah maksimal elemen array.

Contoh pendeklarasian array :

float nilai_tes[5];

menyatakan bahwa array nilai_tes mengandung 5 elemen bertipe float. Pada C, data array akan disimpan dalam memori yang berurutan. Elemen pertama mempunyai indeks bernilai 0. Jika variabel nilai_tes dideklarasikan sebagai array dengan 5 elemen, maka elemen pertama memiliki indeks sama dengan 0, dan elemen terakhir memiliki indeks 4. Gambar di bawah ini menjelaskan urutan komponen dalam array.



Bentuk umum pengaksesan array adalah sbb :

nama_var[indeks]

sehingga, untuk array nilai_tes, maka :

nilai_tes[0] **elemen pertama dari nilai_tes*

nilai_tes[4] **elemen ke-5 dari nilai_tes*

Contoh :

nilai_tes[O] = 7O; / contoh 1 */*
scanf("%f", &nilai_tes[2]); / contoh 2 */*

Contoh pertama merupakan pemberian nilai 70 ke nilai_tes[0]. Sedangkan contoh 2 merupakan perintah untuk membaca data bilangan dari keyboard dan diberikan ke nilai_tes[2]. Pada contoh 2 ini

&nilai_tes[2]

berarti "alamat dari nilai_tes[2]". Perlu diingat bahwa scanf() memerlukan argumen berupa alamat dari variabel yang digunakan untuk menyimpan nilai masukan.

CONTOH PERMASALAHAN 14

```
/*Pemakaian array utk menyimpan sejumlah nilai tes */
#include <stdio.h>
#define MAKS 5
main()
{
int i;
float total = 0, rata;
float   nilai_tes[MAKS];     /* deklarasi array */
for(i=0; i < MAKS; i++)      /* pemasukan data nilai_tes */
{
printf("Nilai tes ke-%d : ", i+1);
scanf("%f", &nilai_tes[i]);
total = total + nilai_tes[i];/* menghitung jumlah seluruh nilai */
}
rata = total / MAKS;     /* hitung nilai rata-rata */
printf("\nNilai rata-rata = %g\n", rata); /* cetak nilai rata-rata */
}
```

Ada beberapa variasi cara mendeklarasikan sebuah array (dalam hal ini yang berdimensi satu), di antaranya adalah sebagai berikut :

- *int numbers[1O];*
- *int numbers[1O] = { 34, 27, 16 };*
- *int numbers[] = { 2, -3, 45, 79, -14, 5, 9, 28, -1, O };*
- *char text[] = "Welcome to New Zealand.";*
- *float radix[12] = { 134.362, 1913.248 };*
- *double radians[1OOO];*

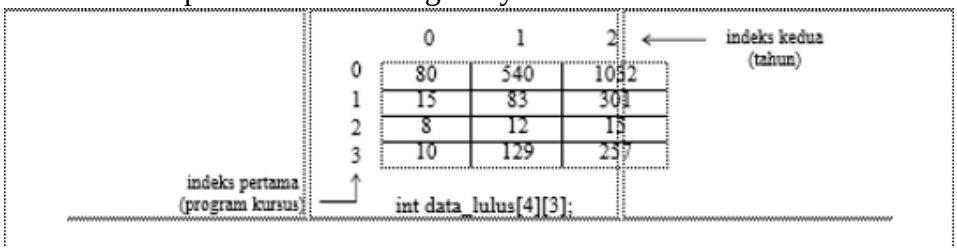
2. Array Berdimensi Dua

Data seperti yang disajikan pada Tabel dibawah ini, dapat disimpan pada sebuah array berdimensi dua. Dimensi pertama dari array digunakan untuk menyatakan kode program kursus dan dimensi kedua untuk menyatakan tahun kursus.

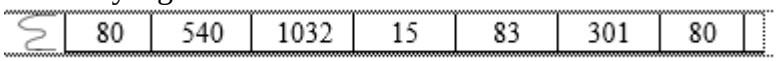
Data Kelulusan Siswa Pada Sebuah Kursus Komputer

Tahun Program	1998	1999	2000
1. (INTRO)	80	540	1032
2. (BASIC)	15	83	301
3. (PASCAL)	8	12	15
4. (C)	10	129	257

Nilai 3 untuk menyatakan banyaknya tahun dan 4 menyatakan banyaknya program kursus. Gambar dibawah ini memberikan ilustrasi untuk memudahkan pemahaman tentang array berdimensi dua.



Sama halnya pada array berdimensi satu, data array aka ditempatkan pada memori yang berurutan. Perhatikan Gambar



Mengakses elemen Array Berdimensi Dua, Array seperti *data_lulus* dapat diakses dalam bentuk

Sama halnya pada array berdimensi satu, data array aka ditempatkan pada memori yang berurutan. Perhatikan Gambardata_lulus[indeks pertama, indeks kedua]

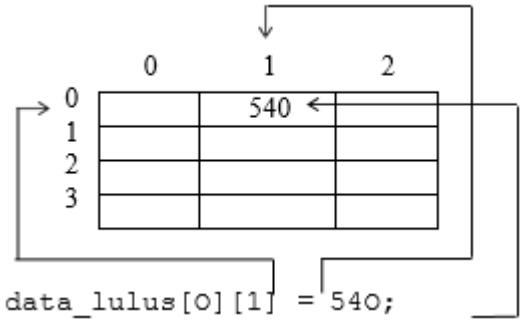
Contoh :

data_lulus[O][1] = 54O;

merupakan instruksi untuk memberikan nilai 540 ke array data_lulus untuk indeks pertama = 0 dan indeks kedua bernilai 1.

printf("%d",data_lulus[2][O]);

merupakan perintah untuk menampilkan elemen yang memiliki indeks pertama = 2 dan indeks kedua = 0.



CONTOH PERMASALAHAN 15

```c
/* File program : lulus.c
Contoh pemakaian array berdimensi dua */
#include <stdio.h>
main( )
{
int tahun, kode_program; int data_lulus[4][3];
/* Memberikan data ke array */
data_lulus[0][0] = 80;
data_lulus[0][1] = 540;
data_lulus[0][2] = 1032;
data_lulus[1][0] = 15;
data_lulus[1][1] = 83;
data_lulus[1][2] = 301;
data_lulus[2][0] = 8;
data_lulus[2][1] = 12;
data_lulus[2][2] = 15;
data_lulus[3][0] = 10;
data_lulus[3][1] = 129;
data_lulus[3][2] = 257;

/* proses utk memperoleh informasi jml siswa yg lulus */
printf("Masukkan tahun dr data yg ingin anda ketahui ");
printf("(1998..2000) : ");
scanf("%d", &tahun);
printf("Masukkan kode program kursus yang ingin anda ketahui");
printf("(1 = INTRO, 2 = BASIC, 3 = PASCAL, 4 = C)   : ");
scanf("%d", &kode_program);
printf("\nTotal kelulusan program tsb = %d\n",data_lulus[kode_program - 1][tahun - 1998] );
}
```

CONTOH PERMASALAHAN 16

```c
#include<stdio.h>
main()
{
    int bilangan [3][2] = {4,6,3,8,5,9};
    int i,j;
    for(i=0; i<3; i++)
    {
    for(j=0; j<2; j++)
    {
        printf("bilangan ke %i,%i=%i\t", i+1, j+1, bilangan [i][j]);
    }
    printf("\n");
    }
return 0;
}
```

B. STRING

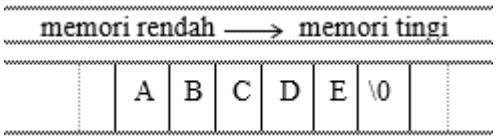
String merupakan bentuk data yang biasa dipakai dalam bahasa pemrograman untuk keperluan menampung dan memanipulasi data teks, misalnya untuk menampung (menyimpan) suatu kalimat. Pada bahasa C, string bukanlah merupakan tipe data tersendiri, melainkan hanyalah kumpulan dari nilai-nilai karakter yang berurutan dalam bentuk array berdimensi satu.

1. Konstansa String

Suatu konstanta string ditulis dengan diawali dan diakhiri tanda petik ganda, misalnya:

"ABCDE"

Nilai string ini disimpan dalam memori secara berurutan dengan komposisi sebagai berikut:



Setiap karakter akan menempati memori sebesar 1 byte. Byte terakhir otomatis akan berisi karakter NULL (0). Dengan mengetahui bahwa suatu string diakhiri nilai NULL, maka akhir dari nilai suatu string akan dapat dideteksi. Sebagai sebuah array karakter, karakter pertama dari nilai string mempunyai indeks ke-0, karakter kedua mempunyai indeks ke-1, dan seterusnya.

2. Inisialisasi String

   Suatu variabel string dapat diinisialisasi seperti halnya array yang lain. Namun tentu saja elemen terakhirnya haruslah berupa karakter NULL. Sebagai contoh :

   char name[] = {'R','I','N', 'I','\O'};

   yang menyatakan bahwa name adalah variabel string dengan nilai awal berupa string : "RINI" . Bentuk inisialisasi yang lebih singkat :

   char name[] = "RINI";

   Pada bentuk ini, karakter NULL tidak perlu ditulis. Secara implisit akan disisipkan oleh kompiler. Perlu diperhatikan, bila name dideklarasikan sebagai string, penugasan (assignment) suatu string ke variabel string seperti

   name = "RINI";

   adalah tidak diperkenankan. Pengisian string ke variabel string akan dibahas pada sub bab berikutnya.

3. I/O Input Output Data String

   Pemasukan data string ke dalam suatu variabel biasa dilakukan dengan fungsi

   gets() atau scanf(). Bentuk umum pemakaiannya adalah sebagai berikut :

```
#include <stdio.h>
gets(nama_array);
```

atau

```
#include <stdio.h>
scanf("%s", nama_array);
```

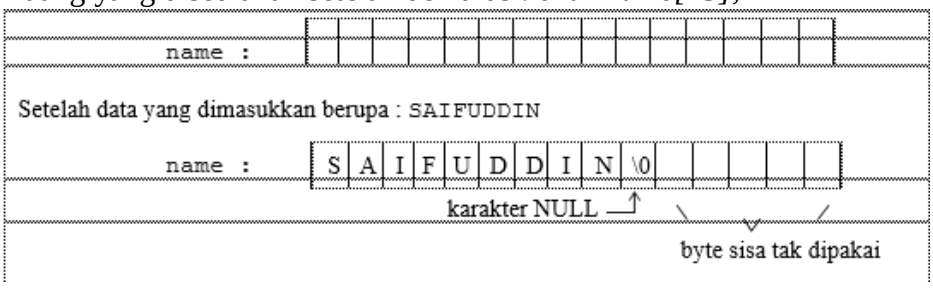
Perhatikan :

nama_array adalah variabel bertipe array of char yang akan digunakan untuk menyimpan string masukan.
Di depan nama_array tidak perlu ada operator & (operator alamat), karena nama_array tanpa kurung siku sudah menyatakan alamat yang ditempati oleh elemen pertama dari array tsb.
Kalau memakai scanf(), data string masukan tidak boleh mengandung spasi.
Di bawah ini diberikan contoh program untuk memasukkan data nama seseorang ke dalam array bernama name.

CONTOH PERMASALAHAN 17

```c
/* File program : yourname.c
Contoh memasukkan data string dari keyboard */
#include <stdio.h>
main()
{
char name[15];

printf("Masukkan nama Anda : ");
scanf("%s", name);

printf("\nHalo, %s. Selamat belajar string.\n", name);
}
```

Ruang yang disediakan setelah deklarasi: *char name[15];*



CONTOH PERMASALAHAN 18

```c
#include <stdio.h>

int main () {
    char  string_1[7] = {'T', 'E', 'K', 'N', 'I', 'K'};
    char  string_2[] = "ELEKTRO";
    printf("Hasil string_1 : %s\n", string_1  );
    printf("Hasil string_2 : %s\n", string_2 );
    getch();

}
```

PRAKTIKUM MINGGU IV

FUNGSI DAN STRUCT

I. Tujuan Praktikum:

1. Mahasiswa dapat memahami kegunaan fungsi
2. Mahasiswa dapat memahami proses deklarasi struct
3. Mahasiswa dapat menerapkan fungsi dan struct

II. Dasar Teori

1. Fungsi

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 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 konstantamaupun ungkapan yang dipakai dalam pemanggilan fungsi. Cara melewatkan/mengirim parameter ke dalam fungsi :
- pengiriman parameter dengan nilai (paramater by value) disebut juga parameter masukan

FUNGSI YANG MENGEMBALIKAN NILAI

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

CONTOH PERMASALAHAN 19

```c
#include <stdio.h>
void info_program()
{
    printf("================================\n");
    printf("Program dibuat oleh Ivon  \n");
    printf("Tanggal : 15 Oktober 2018       \n");
    printf("                                \n");
    printf("Selamat menggunakannya.......   \n");
    printf("================================\n");
}
main()
{
    info_program();
    getchar();
}
```

MACAM FUNGSI

- ✓ Standard, sudah disediakan oleh compiler, tinggal dipakai dengan menyebutkan headernya (kamusnya) pada preprosessor include. Misalnya fungsi printf() headernya adalah stdio.h dan fungsi exit() headernya adalah stdlib.h
- ✓ User, didenisikan oleh user dan disesuaikan dengan kebutuhan user.

Tujuan Fungsi

Fungsi banyak digunakan pada program C dengan tujuan :

- Program menjadi terstruktur, sehingga mudah dipahami dan mudah dikembangkan. Dengan memisahkan langkah-langkah detail ke satu atau lebih fungsi-fungsi, maka fungsi utama (main()) menjadi lebih pendek, jelas dan mudah dimengerti.
- dapat mengurangi pengulangan (duplikasi) kode. Langkah-langkah program yang sama dan dipakai berulang-ulang di program dapat dituliskan sekali saja secara terpisah dalam bentuk fungsi-fungsi. Selanjutnya bagian program yang membutuhkan langkah-langkah ini tidak perlu selalu menuliskannya, tetapi cukup memanggil fungsi-fungsi tersebut.

Dasar Fungsi

Fungsi dasar C yang mengemban tugas khusus contohnya adalah:

- printf(), yaitu untuk menampilkan informasi atau data ke layar.

- scanf(), yaitu untuk membaca kode tombol yang diinputkan.

Pada umumnya fungsi memerlukan nilai masukan atau parameter yang disebut sebagai argumen. Nilai masukan ini akan diolah oleh fungsi.

Hasil akhir fungsi berupa sebuah nilai (disebut sebagai return value atau nilai keluaran fungsi). Oleh karena itu fungsi sering digambarkan sebagai "kotak gelap/black box" seperti ditunjukkan pada gambar di bawah ini.

Fungsi Sebagai Sebuah Kotak Gelap

Parameter bisa diatikan sebagai “bahan baku”yang akan diproses dalam fungsi dan dikirim dari tempat fungsi tersebut dipanggil. Keluaran fungsi (return value) bisa diartikan sebagai “oleh-oleh” yang akan dibawa ketika proses kembali ke tempat asal fungsi tersebut dipanggil. Penggambaran sebagai kotak gelap di antaranya menjelaskan bahwa bagian dalam fungsi bersifat pribadi bagi fungsi. Tak ada suatu pernyataan di luar fungsi yang bisa mengakses bagian dalam fungsi, selain melalui parameter (atau variabel eksternal yang akan dibahas belakangan). Misalnya melakukan goto dari pernyataan di luar fungsi ke pernyataan dalam fungsi adalah tidak diperkenankan. Bentuk umum dari denisi sebuah fungsi adalah sebagai berikut :

```
tipe-keluaran-fungsi nama-fungsi (deklarasi argumen)
{
    tubuh fungsi;
}
```

Keterangan:

tipe-keluaran-fungsi, dapat berupa salah satu tipe data C, misalnya char atau int . Kalau penentu tipe tidak disebutkan maka dianggap bertipe int (secara default).

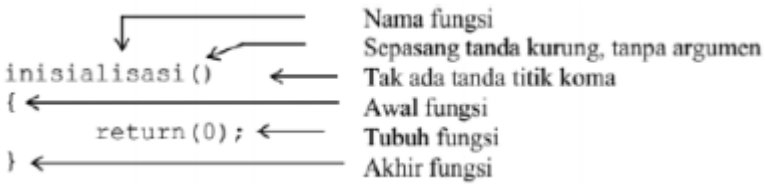
tubuh fungsi, berisi deklarasi variabel (kalau ada) dan statemen-statemen yang akan melakukan tugas yang akan diberikan kepada fungsi yang bersangkutan. Tubuh fungsi ini ditulis di dalam tanda kurung kurawal buka dan kurung kurawal tutup.

Sebuah fungsi yang sederhana bisa saja tidak mengandung parameter sama sekali dan tentu saja untuk keadaan ini deklarasi parameter juga tidak ada. Contoh :

```
int inisialisasi()
{
    return(0);
}
inisialisasi()
{
    return(0);
}
```

Pada fungsi di atas:

- tipe keluaran fungsi tidak disebutkan, berarti keluaran fungsi ber tipe int.
- inisialisasi adalah nama fungsi.
- Tanda () sesudah nama fungsi menyatakan bahwa fungsi tak memiliki parameter.
- Tanda { dan } adalah awal dan akhir fungsi.
- return(0) merupakan sebuah pernyataan dalam tubuh fungsi.



Penjelasan Denisi Sebuah Fungsi

CONTOH PERMASALAHAN 20

```
#include <stdio.h>
int minimum (int, int);
main(){
    int a, b, kecil;
    printf("Masukkan nilai a : ");
    scanf("%d", &a);
    printf("Masukkan nilai b : ");
    scanf("%d", &b);

    kecil = minimum(a, b);
    printf("\nBilangan terkecil antara %d dan %d adalah %d\n\n", a, b, kecil);
}
minimum(int x, int y){
    if (x < y)
    return(x);
    else
}
```

CONTOH PERMASALAHAN 21

```c
#include <stdio.h>

int hitung(int a, int *b){
    *b = 15;
    return a + *b;
}
main(){
    int y,z,hasil;
    y=10;
    z=50;
    printf("Sebelum Jalankan Fungsi: \n ");
    printf("y=%d\n",y);
    printf("z=%d\n",z);
    hasil=hitung(y,&z);
    printf("Setelah Jalankan Fungsi: \n ");
    printf("y=%d\n",y);
    printf("z=%d\n",z);
    printf("hasil=%d\n",hasil);
}
```

2. Struct

   Definisi Struktur yaitu pengelompokan dari variabel-variabel atau sejumlah data dengan tipe yang berlainan yang bernaung dalam satu nama yang sama. Struktur biasa digunakan untuk mengelompokkan beberapa informasi yang berkaitan dengan sebuah kesatuan, atau biasanya disebut dengan record. Cara mendeklarasikan struktur adalah dengan menggunakan kata kunci struct. Perhatikan contoh penulisan struktur berikut ini:

   typedef struct{
   tipe_data <nama_var>;
   tipe_data <nama_var>;
   .... }

   Ada dua cara mendeklarasikan struct pada bahasa C yang pertama yaitu dengan cara:

   a. Menggunakan keyword typedef

      typedef struct Mahasiswa {
      char NIM[12];
      char nama[50];
      float ipk;
      };

   b. Menggunakan Keyword Struct

      struct {
      char NIM[8];
      char nama[50];
      float ipk;
      } mhs;

CONTOH PERMASALAHAN 22

```
#include <stdio.h>

struct Siswa {
  char  nama[50];
  char  sekolah[50];
  unsigned int  uangSaku;
};
int main(void)
{
  struct Siswa siswa01;
  strcpy(siswa01.nama, "Rahendra Putra P");
  strcpy(siswa01.sekolah, "SMK 1 Surabaya");
  siswa01.uangSaku = 10000;
  printf("%s bersekolah di %s \n", siswa01.nama, siswa01.sekolah);
  printf("dengan uang saku %i per hari\n", siswa01.uangSaku);
  return 0;
}
```

CONTOH PERMASALAHAN 23

```
#include <stdio.h>
main()
{
struct zodiak{
    char nama[11];
    int tgl_awal;
    int bln_awal;
    int tgl_akhir;
    int bln_akhir;
};
static struct zodiak bintang = {"Sagitarius", 22, 11, 21, 12};
int tgl_lhr, bln_lhr, thn_lhr;
printf("Masukkan tgl lahir Anda (XX-XX-XXXX): ");
scanf("%d-%d-%d",&tgl_lhr, &bln_lhr, &thn_lhr);
if((tgl_lhr >= bintang.tgl_awal && bln_lhr == bintang.bln_awal)
   || (tgl_lhr <= bintang.tgl_akhir && bln_lhr == bintang.bln_akhir))
printf("Bintang Anda adalah %s\n", bintang.nama);
else
printf("Bintang Anda bukan %s\n", bintang.nama);
}
```

CONTOH PERMASALAHAN 24

```c
#include <stdio.h>

struct mahasiswa {
    char nim[25];
    char nama[25];
    int usia;
};

typedef struct {
    char namamk[25];
    int semester;
    int sks;
}mataKuliah;

void main(){
    struct mahasiswa mhs1 = {"170431100001", "Kina",24};
    mataKuliah mk1 = {"Algoritma dan Pemrograman", 2, 3};
    //tampilkan data Mahasiswa
    printf("NIM : %s\n",mhs1.nim);
    printf("Nama : %s\n",mhs1.nama);
    printf("Usia : %d\n",mhs1.usia);
    //tampilkan data Mata Kuliah
    printf("Mata Kuliah : %s\n",mk1.namamk);
    printf("Semester : %d\n",mk1.semester);
    printf("SKS : %d\n",mk1.sks);
}
```

CONTOH PERMASALAHAN 25

```c
#include <stdio.h>

struct nilai{
    char mataKuliah[25];
    int nilaiMk;
};

struct mahasiswa {
    char nim[25];
    char nama[25];
    struct nilai dataNilai;
};

void main(){
    struct mahasiswa mhs1 = {"170431100001", "Kina",{"Algoritma dan Pemrograman",90}};
    printf("NIM : %s\n",mhs1.nim);
    printf("Nama : %s\n",mhs1.nama);
    printf("Mata Kuliah : %s\n",mhs1.dataNilai.mataKuliah);
    printf("Nilai : %d\n",mhs1.dataNilai.nilaiMk);
}
```

PRAKTIKUM MINGGU V

OPERASI FILE DAN POINTER

I. Tujuan Praktikum:

1. Mahasiswa dapat mengerti tentang operasi file dan input output file
2. Mahasiswa dapat memahami tentang peletakan pointer

II. Dasar Teori

1. Operasi File

   File menurut wikipedia yaitu identitas dari data yang disimpan di dalam berkas sistem yang dapat diakses dan diatur oleh pengguna. Penggunaan dan pengoperasian file pasti selalu dibutuhkan terutama bagi seorang programmer untuk mengolah sebuah data pada file.

   Terdapat 3 mode utama pada file yaitu r (read), w (write), dan a (append). Operasi utama pada file diantaranya membaca, menutup, menghapus, dan mengubah (truncate)

Mode	Fungsi
r	Membaca file (file harus sudah ada)
w	Menulis file (file yang sudah ada akan dihapus)
a	Membuka file yang sudah ada dan pada prosesnya dilakukan penambahan saat menulis (jika file belum ada, otomatis akan dibuat)
r+	Membaca file, tetapi juga memiliki fungsi lain yaitu dapat menulis
w+	Menulis file tetapi juga dapat membaca (file yang sudah ada akan dihapus)

Membuka file :

Untuk membuka sebuah file dibutuhkan sebuah fungsi fopen, directory serta nama file dan mode file, nama variabel file = fopen("nama file", "mode file");. Contohnya :

pf = fopen("data.txt", "r");

CONTOH PERMASALAHAN 26

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

int main(){
    char nama[100];
    int umur;
    FILE *in=fopen("test.txt","r");
        while(!feof(in)){
            fscanf(in,"%[^#]#%d\n", &nama, &umur);fflush(stdin);
            // %[^#] artinya kita menyimpan bagian dari string dalam file sampai tanda #.
            // Kita tidak menggunakan %s karena nama mengandung spasi
            printf("%s %d\n", nama, umur);
        }
    fclose(in);
    getchar();
        return 0;
}
```

CONTOH PERMASALAHAN 27

```
#include <stdio.h>

int main(){
    char nama[100];
        int umur;
        FILE *in=fopen("text.txt","r");
        if(!in){   //cek apakah filenya ada atau tidak
            printf("tidak ada file");
        }else{
            while(!feof(in)){
                fscanf(in,"%[^#]#%d\n", &nama, &umur);fflush(stdin);
                printf("%s %d\n", nama, umur);
            }
            fclose(in);
        }
        getchar();
        return 0;
}
```

Membuat file dan Menulis data ke file :

Untuk membuat sebuah file yang kita perlukan hanyalah menggunakan fungsi FILE yang dilanjutkan pointer nama variabel file tersebut, FILE *nama variabel;. contohnya yaitu :

*FILE *pf;*

fputs(data,fp);

fprintf(fp, "%s", data);

CONTOH PERMASALAHAN 28

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

int main(){
    char nama[100];
    int umur;
    printf("Masukkan nama : "); scanf("%[^\n]", &nama); fflush(stdin);
    printf("Masukkan umur : "); scanf("%d", &umur); fflush(stdin);

    FILE *out=fopen("test.txt","w");
    fprintf(out,"%s#%d\n",nama, umur);
    fclose(out);
    printf("Sukses menambah data.");
    getchar();
        return 0;
}
```

Menambah data ke file :

Append artinya menambahkan data pada file baris terakhir. Jika belum ada data/filenya, maka append akan membuatkan file baru. Contohnya :

CONTOH PERMASALAHAN 29

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

int main(){
    char nama[100];
    int umur;
    printf("Masukkan nama : "); scanf("%[^\n]", &nama); fflush(stdin);
    printf("Masukkan umur : "); scanf("%d", &umur); fflush(stdin);

    FILE *out=fopen("test.txt","a");
    fprintf(out,"%s#%d\n",nama, umur);
    fclose(out);
    printf("Sukses menambah data.");
    getchar();
        return 0;
}
```

2. Pointer

   Pointer adalah sebuah jenis variabel yang dapat menunjuk address atau alamat memory dari sebuah variabel lain. Setiap variabel biasanya mempunyai sebuah alamat karena dalam deklarasi, program akan mengalokasikan sebuah alamat untuk si variabel tersebut. Sebagai contoh kita akan menginisialisasikan variabel angka=10. Bisa dianalogikan variabel angka sebagai rumah, 10 adalah isi dari rumah dan rumah pasti mempunyai alamat. Untuk mendeklarasikan variabel pointer, kita dapat menggunakan simbol bintang (*) di depan variabel yang di deklarasikan pada tipe data tertentu.

```c
#include <stdio.h>
int main(){
    int angka=10;
    int *p;
    p=&angka;
```

```
printf("%d", *p);
getchar();
return 0;
```

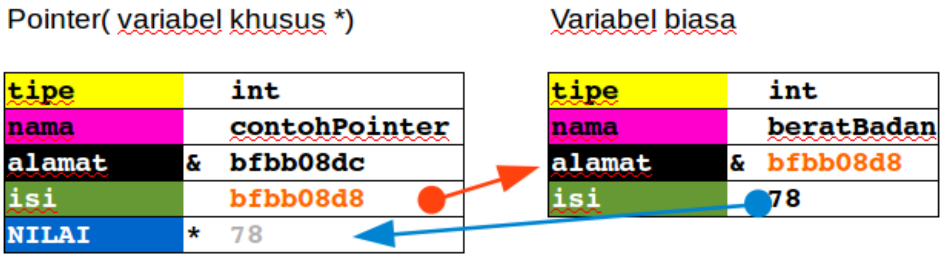
Keterangan :

- Deklarasi awal integer angka diberi nilai 10.
- Deklarasi pointer p (*p)
- p=&angka; artinya kita mengassign atau menunjuk alamat dari variabel angka. Dalam bahasa C, simbol & mempunyai arti address of.
- Untuk mencetak isi dari *p yang mana sudah menunjuk ke alamat variabel angka, maka kita dapat melakukan printf *p; Hasil dari printf *p adalah 10. Karena, diatas kita sudah menunjuk alamat dari variabel angka

Operator Pointer ada dua, yaitu :

1. Operator &(Operator *Deference*)
   - ✓ Operator & bersifat unary (hanya memerlukan satu operand saja)
   - ✓ Operator & menghasilkan alamat dari operandnya.

2. Operator *(Operator *Reference*)
   - ✓ Operator * bersifat unary (hanya memerlukan satu operand saja)
   - ✓ Operator * menghasilkan nilai yang berada pada sebuah alamat.



Setiap variabel memiliki beberapa karakteristik, yaitu :

1. tipe
2. nama
3. alamat
4. isi/NILAI

Khusus untuk pointer memiliki karakteristik tambahan

1. tipe
2. nama
3. alamat
4. isi
5. NILAI

Bedanya adalah bahwa pada variabel biasa isi dan NILAI dapat dianggap sama. Sedangkan '/pointer membedakan isi dengan NILAI, isi adalah isi memory

secara fisik, yaitu (menuju) alamat dari varibel lain, sedangkan NILAI adalah isi dari variabel yang dituju.

Contoh program fungsi dengan parameter pass by reference

```
#include <stdio.h>

int hitung(int a, int *b){
    *b = 15;
    return a + *b;

}

main(){
    int y,z,hasil;
    y=10;
    z=50;
    printf("Sebelum Jalankan Fungsi: \n ");
    printf("y=%d\n",y);
    printf("z=%d\n",z);
    hasil=hitung(y,&z);
    printf("Setelah Jalankan Fungsi: \n ");
    printf("y=%d\n",y);
    printf("z=%d\n",z);
    printf("hasil=%d\n",hasil);
}
```

CONTOH PERMASALAHAN 30

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int I=88,F;
    int*IF;
    F=I;//nilai variabel F di isi nilai variabel I
    IF=&I;//pointer IF menuju alamat memori variabel I
    //sekarang alamat memori variabel I dan IF sama
    *IF=77;//alamat memori pointer IF di isi 16, sekarang IF bernilai 16
    //pengisian nilai pointer IF langsung melalui alamat memorinya
    printf("nilai dari IF=%d\n",*IF);
    printf("nilai dari I =%d\n",I);
    printf("nilai dari F =%d\n",F);
    return 0;
}
```

CONTOH PERMASALAHAN 31

```c
#include <stdio.h>

main(){
    int *ptr;
    int k;
    k=7;
    printf("Isi variabel k = %d",k);
    printf("\nAlamat variabel k = %d",&k);
    printf("\nAlamat variabel *ptr = %d",&ptr);
    printf("\nIsi variabel *ptr = %d",ptr);
    ptr=&k;
    printf("\nAlamat variabel *ptr = %d",&ptr);
    printf("\nIsi variabel *ptr = %d",ptr);
    printf("\nIsi dari alamat %d = %d",ptr,*ptr);
    printf("\n");
}
```

Pointer dan Array

Hubungan antara pointer dan array pada C sangatlah erat. Sebab sesungguhnya array secara internal akan diterjemahkan dalam bentuk pointer. Pembahasan berikut akan memberikan gambaran hubungan antara pointer dan array. Misalnya dideklarasikan di dalam suatu fungsi

static int tgl_lahir[3] = { 01, 09, 64 };

dan

*int *ptgl;*

Kemudian diberikan instruksi

ptgl = &tgl_lahir[0]; //pointer to array of integer

maka ptgl akan berisi alamat dari elemen array tgl_lahir yang berindeks nol. Instruksi di atas bisa juga ditulis menjadi

ptgl = tgl_lahir;

sebab nama array tanpa tanda kurung menyatakan alamat awal dari array. Sesudah penugasan seperti di atas,

**ptgl*

```
//Program berikut ini menampilkan alamat memory dan mengakses larik
menggunakan pointer.
#include <stdio.h>

main(){
    int *p_array;
    int my_array[6] = {1,23,17,4,-5,100};

     // Dapat juga ditulis  p_array = my_array;
    p_array = &my_array[0];

    printf("Alamat dari p_array=%d \n",p_array);
    printf("Isi dari p_array=%d \n",*p_array);

    p_array++;
    printf("Alamat dari p_array=%d \n",p_array);
    printf("Isi dari p_array=%d \n",*p_array);

    p_array++;
    printf("Alamat dari p_array=%d \n",p_array);
    printf("Isi dari p_array=%d \n",*p_array);

    p_array++;
    printf("Alamat dari p_array=%d \n",p_array);
    printf("Isi dari p_array=%d \n",*p_array);
}
```

Program berikut ini menampilkan larik dengan menggunakan pointer.

```
#include <stdio.h>

main(){

    int *pArray, Array[10],i;
    for(i=0;i<10;i++){
        Array[i] = i+10; //pengisian array
    }
    // tunjuk pArray ke alamat awal array
    pArray = &Array[0]; // bisa dituliskan pArray=Array
    for(i=0;i<10;i++){
        printf("Alamat pointer= %d. Isi dari alamat %d =
%d\n",&pArray,pArray,*pArray++); //cetak pArray
    }
}
```

CONTOH PERMASALAHAN 32

```
#include <stdio.h>
main(){
    static int tgl_lahir[] = {16, 4, 1974}; int *ptgl;

    ptgl = tgl_lahir;

    printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
    printf("Nilai dari tgl_lahir[0] = %d\n", tgl_lahir[0]);
}
```

CONTOH PERMASALAHAN 33

```c
#include <stdio.h>

void rubah(int *b){
     printf("%d\n",b);
    *--b = 7;
}

main(){
    int my_array[6] = {1,23,17,4,-5,100};
    int i;
    printf("Menampilkan Data Array\n");
    //Menggunakan Looping
    for(i=0;i<6;i++){
        printf("Data ke-%d =  %d\n",i+1,my_array[i]);
    }
    rubah(&my_array[3]);
    printf("Menampilkan Data Array Setelah di rubah\n");
    for(i=0;i<6;i++){
        printf("Data ke-%d =  %d\n",i+1,my_array[i]);
    }
}
```

CONTOH PERMASALAHAN 34

```c
#include <stdio.h>

main(){
    int *ptr,i,nilai,arrayA[3];
    ptr=arrayA;
    for(i=0;i<3;i++){
        printf("Isi nilai[%d] = ",i);
        scanf("%d",&nilai);
        *ptr=nilai;
        ptr++;
    }
    for(i=0;i<3;i++){
        printf("Isi nilai[%d] = %d",i,arrayA[i]);
        printf("\n");
    }
    printf("\n");
}
```

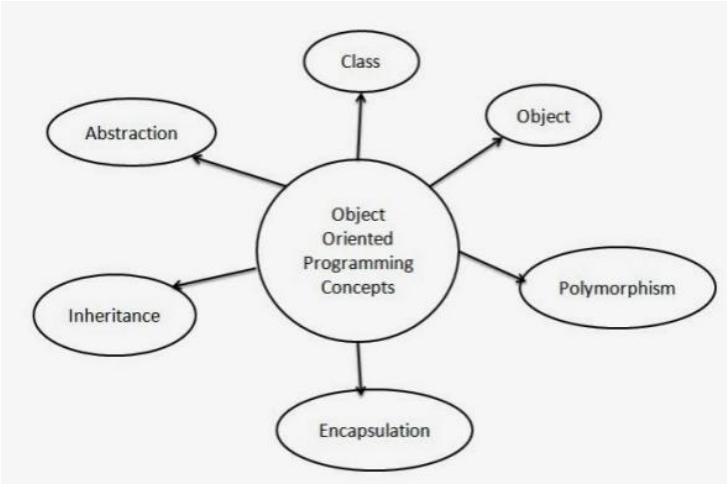
PRAKTIKUM MINGGU VI

OBJECT ORIENTED PROGRAM (C++)

I. Tujuan Praktikum:

1. Mahasiswa dapat mengenal tentang Object Oriented Programming.
2. Mahasiswa dapat mengerti dan menerapkan Konstruktor, Destruktor dan Inheritance.

II. Dasar Teori



Object Oriented Programming atau disingkat menjadi OOP adalah paradigma pemrograman dalam melakukan pemrograman yang beriontasi kepada

object, semua fungsi, data dan pengolahan data akan dibungkus dalam kelas-kelas dan object-object.

Masing-masing object dapat memiliki sifat dan tugasnya. Pada paradigm ini, object-object tersebut dapat berkerja sendiri dan juga dapat saling bekerja sama dengan kemungkinkan untuk saling berhubungan, seperti menerima, mengirim data kepada object lainnya dan memproses data.

Jenis-Jenis OOP pada Bahasa Pemrograman

Karena kemudahan yang diberikan oleh konsep OOP, banyak Bahasa yang membawa dukungan fitur OOP, karena hal itu OOP menjadi memiliki dan dibagi menjadi 3 jenis, yaitu :

- Bahasa OOP Murni adalah sebuah bahasa yang mengharuskan program ditulis hanya berupa object saja. Contoh – Eifel, Smaltalk, Ruby, Jade dan lain-lain.
- Bahasa OOP Hybrid adalah bahasa yang dirancang untuk pemrograman object dengan beberapa elemen procedural.
- Bahasa OOP Hybrid dalam Web sama seperti Bahasa OOP Hybrid, yang berbeda hanyalah, konsep ini sering digunakan dalam pemrograman Web.

Konsep Object Oriented Programming

- Class (Kelas) adalah sebuah rancangan (mirip seperti struct) untuk mendefinisikan karakter dan perilaku dari object. yang merupakan kumpulan atas definisi dan fungsi-fungsi dalam suatu unit, untuk suatu tujuan tertentu.
- Object atau instance adalah dasar dari modularitas dan structur pada OOP. dan merupakan representasi dari class, object akan memiliki sifat dan perilaku dari class yang digunakan.
- Encapsulation adalah konsep dalam implementasi untuk membungkus data dan fungsi menjadi satu entitas, dan membatasi akses dari luar class.
- Inheritance adalah konsep pewarisan class. Class juga dapat menuruni dan memiliki apa yang dimiliki oleh class lainnya.
- Abstraction adalah konsep untuk mendisain sebuah object, teknik dalam menyembunyikan detail suatu proses dalam object tersebut. dengan tujuan untuk memfokuskan pengguna pada fungsi inti object.
- Polymorphism berasal dari bahasa yunani yang berarti memiliki banyak bentuk, adalah kemampuan dalam menyampaikan pesan tertentu keluar dari hirarki objectnya, dimana object yang berbeda memberikan tanggapan atau respon terhadap pesan yang sama sesuai dengan sifat masing-masing object.

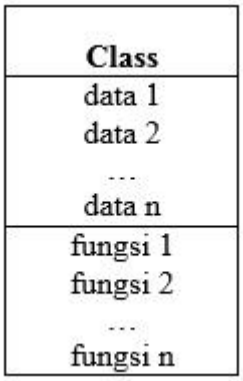
A. Pengenalan Class

Class merupakan blueprint (cetak biru) untuk menciptakan suatu instance dari objek dimana terdiri dari sekumpulan objek dengan kemiripan data / properties / attributes, fungsi / behavior / method dan relasi ke objek lain. Permograman C++ memungkinkan pembuatan class lebih dari 1. Ketika data dan fungsi yang terkait disimpan di dalam sebuah class mampu membantu mem-visualisasikan permasalahan yang kompleks dengan efisien dan efektif.

Contoh: ClassMahasiswa, ClassDosen, Class Flowers.

- **nama_class :** tempat dimana anda dapat memberikan nama pada *class* tersebut.
- **akses_specifier :** tempat dimana anda dapat mendifinisikan hak akses kepada anggota yang ada dibawahnya. Jika tidak didefinisikan maka anggota *class* secara otomatis memiliki hak akses ***private***.
- **data_member :** tempat dimana anda dapat mendirikan sebuah variabel atau function sebagai anggota dari class.
- **nama_object :** tempat dimana anda dapat mendirikan object-object dengan menggunakan class tersebut, hal ini merupakan opsional tapi jika class tidak diberi nama class maka diwajibkan mendirikan object, karena kita tidak akan bisa membuat object dengan class di luar dari deklarasi class tersebut.

Class dan *Object* merupakan fitur yang sangat membantu untuk mendirikan sebuah program besar, menjadikan sebuah code program yang ditulis oleh programmer mudah untuk dimengerti, lebih terstruktur, dan juga mudah dalam pemeliharaan program.



Berdasarkan Gambar diatas, data dan fungsi yang berada di dalam sebuah classdisebut sebagai anggota dari suatu class. Data pada suatu class digunakan untuk memegang informasi yang ada pada class tersebut, sedangkan fungsi digunakan sebagai behavior dari class tersebut.
Untuk pembuatan sebuah class, dimulai dengan kata kunci class dan diikuti dengan nama kelasnya, dibuka dengan{, isidariclass, ditutupdengan};. Berikut adalah sintaks pembuatan class:

```
Classclass_name
{
    Data Members;
    Methods;
};
```

Contoh:

```
classBox{

public:

      double length; // Panjang box

      double width;  // Lebar box

      double height; // Tinggi box

};
```

Seperti yang telah disebutkan di atas, pendefinisian suatu kelas dimulai dengan kata kunci class dan diikuti dengan nama kelas Box dalam kasus ini. Isi dari suatu kelas ditandai dengan { dan diakhiri dengan } diikuti;. Kata kunci public pada contoh di atas menentukan cara pengaksesan anggota kelas. Anggota kelas public dapat diakses di kelas manapun.Terdapat beberapa jenis pengaksesan anggota kelas lainnya (access specifier) yang akan dibahas pada bab Encapsulation.

B. Member Class

Seperti yang telah disinggung sebelumnya bahwa data dan fungsi di dalam suatu class disebut dengan anggota suatu class.Perhatikan contoh berikut ini:

```
classBox{
public:
double length; // Length of a box
double breadth; // Breadth of a box
double height; // Height of a box

void print()
{
       cout<<"Printing Box Object"<<endl;
}
};
```

Berdasarkan contoh di atas, length, breadth dan height merupakan data member dari class Box; sedangkan print merupakan function member (member fungsi) dariclassBox.

C. Pengenalan Object

Jika class menyediakan blueprint untuk membuat objek, maka, secara dasarnya, objek dibentuk dari suatu class. Pada intinya, objek adalah suatu kumpulan yang memiliki atribut dan metode yang sama (instance dari class). Dalam konteks variabel, suatu class dapat dianggap sebagai tipe data, dan objek sebagai variablenya.

Contoh: Dari class Flowers dapat dihasilkan objek Rose, Orchid, SunFlower, dsb.

Sintaks untuk membuat sebuah objek ialah:

```
class_name variable name;
```

Berdasarkan contoh kelas Box di atas, maka objeknya ialah:

```
Box obj1, obj2;
```

Berdasarkan contoh di atas, terdapat dua buah objek yang berasal dari kelas Box, yaitu obj1 dan obj2.

D. Mengakses Data Member dan Function Member

Datamember dan memberfunction (anggota data dan fungsi) dapat diakses dengan menggunakan operator (.). Secara umum, sintaks untuk mengakses datamemberataupun functionmember ialah:

```
object_name.data_member;
```

CONTOH PERMASALAHAN 35

```
#include<iostream>
using namespace std;

class oop{
    public :
    int a,b,c;
};
main(){
    komb14 oop;
    cout<<"bil 1 : ";
    cin>>oop.a;
    cout<<"bil 2 : ";
    cin>>oop.b;
    cout<<"bil 3 : ";
    cin>>oop.c;
    cout<<"Hasil = "<<oop.a+oop.b+oop.c;
    cout<<endl;
}
```

Konstruktor dan Destruktor

Konstruktor adalah fungsi khusus anggota kelas yang otomatis dijalankan pada saat penciptaan objek (mendeklarasikan instance). Konstruktor ditandai dengan namanya, yaitu sama dengan nama kelas. Konstruktor tidak mempunyai tipe hasil, bahkan juga bukan bertipe void. Biasanya konstruktor dipakai untuk inisialisasi anggota data dan melakukan operasi lain seperti membuka file danmelakukan alokasi memori secara dinamis. Meskipun konstruktor tidak harus ada di dalam kelas, tetapi jika diperlukan konstruktor dapat lebih dari satu.

Tiga jenis konstruktor:

1. Konstruktor default : tidak dapat meneriman argumen, anggota data diberi nilai awal tertentu
2. Konstruktor penyalinan dengan parameter : anggota data diberi nilai awal berasal dari parameter.
3. Konstruktor penyalinan objek lain : parameter berupa objek lain, anggotadata diberi nilai awal dari objek lain.

Perhatikan contoh berikut:

```
//Program Konstruktor
#include<iostream.h>
#include<conio.h>
class titik
{
        int x;
        int y;
public:
        titik()                                 //konstruktor
default
        {
                x=0;
                y=0;
        }

        titik(int nx, int ny)    // konstruktor penyalinan
        {
                x=nx;
                y=ny;
        }
        titik(const titik& tt)   // konstruktor penyalinan objek
        {
                x=tt.x;
                y=tt.y;
        }
        int NX() { return x; } // fungsi anggota biasa
        int NY() { return y; } // fungsi anggota biasa
};


void main()
{
        titik t1;  // objek dg konstruktor default
        titik t2(10, 20); // objek dg konstruktor penyalinan
        titik t3(t2); // objek dg konstruktor penyalinan objek
        cout<<"t1 = "<< t1.NX() << ", "<<t1.NY()<<endl;
        cout<<"t2 = "<< t2.NX() << ", "<<t2.NY()<<endl;
        cout<<"t3 = "<< t3.NX() << ", "<<t3.NY()<<endl;
        getch();
}
```

Objek t1 diciptakan dengan otomatis menjalankan konstruktor default. Dengan demikian anggota data diberi nilai x=0, dan y = 0. Penciptaan objek t2 diikuti dengan menjalankan konstruktor kedua, mengakibatkan anggota data diberi nilai x=10 dan y=20. Terakhir, t3 diciptakan dengan menyalin objek dari t2, mengakibatkan pemberian nilai kepada anggota data sama dengan objek t2.

Destruktor

Destruktor adalah pasangan konstruktor. Pada saat program menciptakan objek secara otomatis konstruktor akan dijalankan, yang biasanya dimaksudkan untuk memberi nilai awal variabel private. Sejalan dengan ini, C++ menyediakan fungsi destruktor (penghancur atau pelenyap) yang secara otomatis akan dijalankan pada saat berakhirnya kehidupan objek. Fungsi destruktor adalah untuk mendealokasikan memori dinamis yang diciptakan konstruktor. Nama destructor sama dengan nama kelas ditambah awalan karakter tilde(~). Perhatikan contoh berikut:

```
// Contoh Konstruktor dan Destruktor
#include<iostream.h>
class Tpersegi
{
        int *lebar, *panjang;
public:
        Tpersegi (int, int);
        ~Tpersegi();
        int Luas() {return (*lebar * *panjang);}
};

Tpersegi::Tpersegi(int a, int b)
{
        lebar = new int;
        panjang = new int;
        *lebar = a;
        *panjang = b;
}

Tpersegi::~Tpersegi()
{
        delete lebar;
        delete panjang;
}

int main()
{
        Tpersegi pers(3,4), persg(5,6);
        cout<< "Luas pers = "<<pers.Luas()<<endl;
        cout<< "Luas persg = "<<persg.Luas()<<endl;
        return 0;
}
```

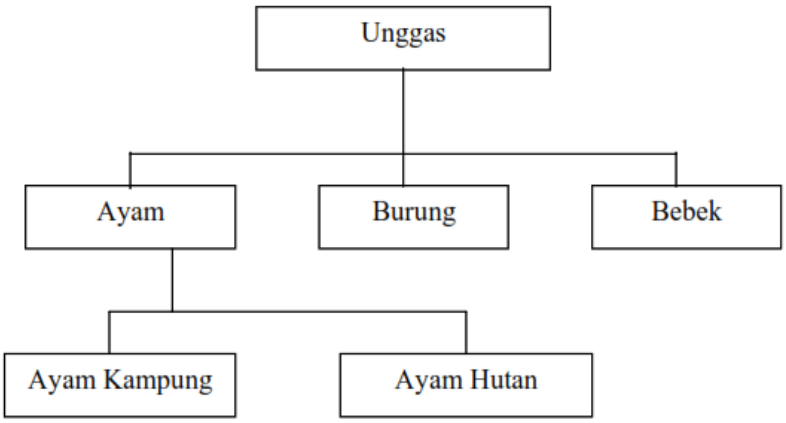
Pewarisan (Inheritance)

Suatu kelas dapat diciptakan berdasarkan kelas lain. Kelas baru ini mempunyai sifat-sifat yang sama dengan kelas pembentuknya, ditambah sifat-sifat khusus lainnya. Dengan pewarisan kita dapat menciptakan kelas baru yang mempunyai sifat sama dengan kelas lain tanpa harus menulis ulang bagian-bagian yang sama. Pewarisan merupakan unsur penting dalam pemrograman berorientasi objek dan merupakan blok bangunan dasar pertama penggunaan kode ulang (code reuse).

Jika tidak ada fasilitas pewarisan ini, maka pemrograman dalam C++ akan tidak banyak berbeda dengan pemrograman C, hanya perbedaan dalam pengkapsulan saja yang menggunakan kelas sebagai pengganti struktur. Yang perlu menjadi catatan di sini adalah bahwa data dan fungsi yang dapat diwariskan hanya yang bersifat public dan protected. Untuk data dan fungsi private tetap tidak dapat diwariskan. Hal ini disebabkan sifat protected yang hanya dapat diakses dari dalam kelas saja.

Sifat pewarisan ini menyebabkan kelas-kelas dalam pemrograman berorientasi objek membentuk hirarki kelas mulai dari kelas dasar, kelas

turunan pertama, kelas turunan kedua dan seterusnya. Sebagai gambaran misalnya ada hirarki kelas unggas.



Sebagai kelas dasar adalah Unggas. Salah satu sifat Unggas adalah bertelur dan bersayap. Kelas turunan pertama adalah Ayam, Burung dan Bebek. Tiga kelas turunan ini mewarisi sifat kelas dasar Unggas yaitu bertelur dan bersayap. Selain mewarisi sifat kelas dasar, masing-masing kelas turunan mempunyai sifat khusus, Ayam berkokok, Burung terbang dan Bebek berenang. Kelas Ayam punya kelas turunan yaitu Ayam Kampung dan Ayam Hutan. Dua kelas ini mewarisi sifat kelas Ayam yang berkokok. Tetapi dua kelas ini juga punya sifat yang berbeda, yaitu: Ayam Kampung berkokok panjang halus sedangkan Ayam hutan berkokok pendek dan kasar. Jika hirarki kelas Unggas diimplementasikan dalam bentuk program, maka secara sederhana dapat ditulis sebagai berikut:

```
//Program Kelas Unggas
#include<iostream.h>
#include<conio.h>
class Unggas
{
public:
        void Bertelur() {cout<<"Bertelur"<<endl; }
};

class Ayam : public Unggas
{
public:
        void Berkokok() {cout<<"Berkokok"<<endl; }
};

class Burung : public Unggas
{
public:
        void Terbang() {cout<<"Terbang"<<endl; }
};

class Bebek : public Unggas
{
public:
        void Berenang() {cout<<"Berenang"<<endl; }
};

class AyamKampung : public Ayam
{
public:
        void Berkokok_Panjang_Halus() {cout<<"Berkokok Panjang Halus"<<endl; }
};
```

```
class AyamHutan : public Ayam
{
public:
        void Berkokok_Pendek_Kasar() {cout<<"Berkokok Pendek Kasar"<<endl; }
};

void main()
{
        cout <<"Sifat bebek adalah:"<<endl;
        Bebek bk;
        bk.Bertelur();
        bk.Berenang();
        cout<<endl;

        cout<<"Sifat ayam adalah:"<<endl;
        Ayam ay;
        ay.Bertelur();
        ay.Berkokok();
        cout<<endl;

        cout<<"Sifat ayam kampung adalah:"<<endl;
        AyamKampung ayk;
        ayk.Bertelur();
        ayk.Berkokok();
        ayk.Berkokok_Panjang_Halus();
        getch();
}
```

Dapat dilihat, bahwa kelas Ayam dan kelas Bebek dapat menjalankan fungsi Bertelur() yang ada dalam kelas Unggas meskipun fungsi ini bukan merupakan anggota kelas Ayam dan kelas Bebek. Kelas AyamKampung dapat menjalankan fungsi Berkokok() yang ada dalam kelas Ayam walaupun dua fungsi tersebut bukan merupakan anggota kelas AyamKampung. Sifat-sifat di atas yang disebut dengan pewarisan (inheritance).

CONTOH PERMASALAHAN 36

```
#include <iostream>
using namespace std;
#include <conio.h>

class Mobil
{
private :
      int roda, pintu;
      char warna [10];
public :
      void data(int jum_roda, int jum_pintu, char warna_mobil [10]);
      void info();
};
int main()
{
      //mendeklarasikan objek
      Mobil lamborghini;
      Mobil bmw;

      cout<<"Mobil Lamborghini :"<<endl;
      cout<<"--------------------"<<endl;
      lamborghini.data(4,2,"Hitam"); //memanggil fungsi data()
      lamborghini.info(); //memanggil fungsi info();

      cout<<endl;
      cout<<"Mobil BMW :"<<endl;
      cout<<"-----------"<<endl;
      bmw.data(4,4,"Silver"); //memanggil fungsi data()
      bmw.info(); //memanggil fungsi info();
      _getche();
      return 0;
}
void Mobil::data(int jum_roda, int jum_pintu, char warna_mobil [10])
{
      roda  = jum_roda;
      pintu = jum_pintu;
      strcpy(warna, warna_mobil);
}

void Mobil::info()
{
      cout<<"Jumlah Roda  : "<<roda<<endl;
      cout<<"Jumlah Pintu : "<<pintu<<endl;
      cout<<"Warna        : "<<warna<<endl;
}
```

CONTOH PERMASALAHAN 37

```
#include<iostream>
using namespace std;
class jajargenjang{
private :
    int sisisejajar;
    int tinggi;
    int luas;

public :
    void input ( int m, int n{
        sisisejajar=m;
        tinggi=n;
        luas=sisisejajar+sisisejajar*tinggi/2;
    }
    void hasilluasjajargenjang(){
        cout<<luas;
    }
};

int main()
{
    int m;
    int n;
    int luas;
    cout<<"\t\t\tPROGRAM LUAS JAJAR GENJANG"<<endl;
    cout<<"\t\t\t PRAKTIKUM ALPRO 2019"<<endl;
    cout<<endl;
    cout<<"-------------------------------------------------------------"<<endl;
    jajargenjang jajar;
    cout<<"input sisi\t: ";
    cin>>m;
    cout<<"input tinggi\t: ";
    cin>>n;
    cout<<endl;
    luas = (m+m)*n/2;
    cout<<"\t\tLuas jajar genjang adalah : "<<luas;
    cout<<endl<<endl;
}
```