ALUR PROGRAM

SELEKSI KONDISI
Statement If

a. Bentuk If tunggal sederhana

Sintaks :

```
if ( kondisi ) statement ;
```

Bentuk ini menunjukkan jika kondisi bernilai benar, maka statement yang mngikutinya akan di-eksekusi. Jika tidak maka statement selanjutnya yang akan diproses.

b. Bentuk If tunggal blok statement

Sintaks :

```
if ( kondisi ) {
        blok statement;
        }
```

Perbedaan dengan bentuk sebelumnya statement yang akan dilaksanakan ada dalam satu blok kurung kurawal.

c. Bentuk If..Else

sintaks :

```
if ( kondisi )
        statement1;
else
        statement2;
```

Statement setelah kondisi atau statement sesudah else dapat berupa statement kosong, statement tunggal maupun blok statement. statement1 akan dijalankan jika kondisi benar, jika salah maka statement2 yang akan diproses.

contoh :

```
//Program menentukan ganjil atau genap
#include<stdio.h>

int main(){
  int Bilangan;
  char Lagi;
    printf("Mencari Bilangan Ganjil atau Genap\n\n");
    printf("Input Bilangan :  ");
    scanf("%d", &Bilangan);
    if(Bilangan %2 == 1)
                printf("\n\nIni Bilangan Ganjil");
                else
                        printf("\n\nIni Bilangan Genap");
    return 0; }
```

Output :

```
Mencari Bilangan Ganjil atau Genap
Input Bilangan :  15
Ini Bilangan Ganjil
```

d. Bentuk If..else if…else

Sintaks :

```
if ( kondisi 1)
        statement1;
else if ( kondisi 2 )
        statement2;
else if ( kondisi 3)
        statement3;
        .
else
        statement default;
```

Proses akan mulai dari penyeleksian kondisi 1, jika benar maka statement yang mengikutinya akan dieksekusi, jika salah maka akan masuk proses seleksi kondisi 2, begitu seterusnya. Jika semua kondisi tidak ada yang terpenuhi, maka program akan menjalankan statement default.

contoh :

```
//Program Mencari Mutu Nilai
#include<stdio.h>

int main(){
  int Nilai; char Mutu;
  printf("Mencari Mutu Nilai\n\n");
  printf("Input Nilai Mahasiswa :  ");scanf("%d", &Nilai);
  if (Nilai<50) Mutu = 'E';
          else if(Nilai<65) Mutu = 'D';
                  else if(Nilai<75) Mutu ='C';
                          else if (Nilai<85) Mutu ='B';
                                  else Mutu = 'A';
  printf("\n\nNilai Mahasiswa yang diinput  =  %d", Nilai);
  printf("\nMutu Nilai = %c", Mutu);
  return 0; }
```

Output :

```
Mencari Mutu Nilai
Input Nilai Mahasiswa :  78
Nilai Mahasiswa yang diinput  =   78
Mutu Nilai =  B
```

e. Bentuk If bersarang (nested if)

Sintaks :

```
if ( kondisi 1)
     if ( kondisi 2)
        .
        .
        if (kondisi n )
                statement;
        else
                statement;
             .
             .
           else
                statement
     else statement;
```

Kondisi yang akan diseleksi pertama kali adalah kondisi yang paling luar (kondisi 1). Jika bernilai tidak benar maka statement setelah else yang terluar (pasangan dari if yang bersangkutan) yang akan diproses.

f. Bentuk If dengan kondisi berupa variable

Contoh :

```
if ( D == 0 )
        printf (“Nilai D sama dengan Nol \n”);
else
        printf (“Nilai D tidak sama dengan Nol \n”);
```

g. Bentuk If dengan kondisi Jamak

Beberapa kondisi dapat diseleksi sekaligus dalam statement if dengan menggunakan operator logika AND (&&), OR (||), atau NOT (!)

h. Operator ?

Dapat digunakan untuk menggantikan statement if..else..

sintaks :

```
( kondisi ) ? statement1 : statement2;
```

jika benar statement1 akan diproses, jika salah statement2 yang akan diproses.

Statement Switch

a. Statement Switch tunggal

Sintaks :

```
switch ( kondisi ) {
case konstanta1  :
        statement-statement;
        break;
case konstanta2  :
        statement-statement;
        break;
  .
  .
default  :
        statement-statement;
}
```

contoh :

```
//Program dengan switch Case
#include<stdio.h>

int main(){
   int Pilih;
   printf("----MENU BUAH----\n");
   printf("\n1.  APEL");
   printf("\n2.  MANGGA");
   printf("\n3.  JERUK");
   printf("\n4. KELUAR");
   printf("\n\nPilihan Anda  [1-4]  : ");
   scanf("%d",&Pilih);
   switch(Pilih){
     case 1 : printf("\n\nANDA PILIH APEL"); break;
```

```
    case 2 : printf("\n\nANDA PILIH MANGGA"); break;
    case 3 : printf("\n\nANDA PILIH JERUK"); break;
    case 4 : exit(0);
   default : printf("\n\nANDA SALAH INPUT...");
  }
 return 0; }
```

b. Statement nested switch

Yaitu statement switch yang berada didalam switch lainnya.

Sintaks :

```
switch ( kondisi ) {
case konstanta 1  :
        statement-statement ;
        switch ( kondisi x ) {
        case  konstanta 1a  :
                statement-statement ;
                break;
        case konstanta 1b  :
                statement-statement ;
                break;
        }
        break;
case konstanta 2  :
        statement-statement;
        break;
}
```

PERULANGAN

a. Statement for

Sintaks :

```
for ( inisialisasi; terminasi; iterasi ) statement;
```

inisialisasi adalah pemberian nilai awal variable untuk perulangan, *terminasi* adalah pemberian nilai akhir atau batas perulangan, *iterasi* adalah perubahan variable kontrol (counter).

b. Statement while

Sintaks :

```
while (kondisi ) statement;
```

Statement dapat berupa statement kosong, statement tunggal maupun blok statement. Proses perulangan akan terus dilaksanakan jika kondisi dalam while masih bernilai benar.

c. Statement do…while

Sintaks :

```
do
        statement
while   ( kondisi )
```

Sedikitnya statement akan diproses sebanyak 1 kali karena seleksi kondisi dilaksanakan diakhir statement.

d. Statement continue

Statement continue akan menyebabkan proses perulangan kembali ke awal perulangan dengan mengabaikan statement setelah statement continue.

contoh :

```
//Program dengan for & continue
#include<stdio.h>

int main(){
        int X;
        for (X=0; X<10; X++){
                if (X==5) continue;
                printf("%d  ", X);
        }
        return 0;}
```

Output : 0 1 2 3 4 6 7 8 9

e. Statement break

Statement break akan menyebabkan proses keluar dari blok looping atau blok statement pada case.

contoh :

```
//Program dengan for & break
#include<stdio.h>

int main(){
        int X;
        for (X=0; X<10; X++){
                if (X==5) break;
                printf("%d  ", X);
        }
        return 0;}
```

Output : 0 1 2 3 4

f. Statement goto label

Digunakan untuk melompat dari satu proses ke proses tertentu didalam program.

Sintaks :

```
goto label;
```

Proses lain yang ditunjuk sebagai lompatan akan ditulis

```
label :
```

ARRAY(LARIK)

Larik adalah kumpulan nilai-nilai data bertipe sama dalam urutan tertentu yang menggunakan sebuah nama yang sama. Nilai-nilai data di suatu larik disebut dengan elemen larik yang letak urutannya ditunjukkan oleh suatu *subscript* atau suatu *index* yang dimulai dengan index nol.

Larik dapat berdimensi satu (one dimensional array) yang mewakili suatu vektor, larik berdimensi dua (two dimensional array) mewakili bentuk suatu matrik atau tabel, larik berdimensi tiga (three dimensional array) mewakili suatu bentuk ruang atau berdimensi lebih dari tiga.

Array Dimensi 1

Suatu larik dapat dideklarasikan dengan menyebutkan jumlah dari elemennya yang dituliskan diantara tanda '[ ]'. Contoh :

```
Int X[5];
```

Berarti variabel X bertipe integer dan merupakan larik dimensi satu.

Contoh larik berdimensi Satu :

```
#include <stdio.h>
main()
{
        float X[3] = {5,3,7}, Total = 0;
        int I;
        for(I=0;I<=2;I++)  Total = Total + X[I];
        printf('Total = %f \n",Total);
        }
```

Output :
Total = 15.000000

Array Dimensi 2

Pendeklarasian larik dimensi dua :

int X[3][4]; → berarti akan membentuk matrik dengan ukuran 3 baris X 4 kolom

int X[2][3]={1,2,3,4,5,6} → matrik 2X3

atau larik tidak berukuran seperti :

int X[ ][4] ={1,2,3,4,5,6,7,8} → matrik 2X4

Contoh larik dimensi dua :

```
#include <stdio.h>
main(){
   int I,J;
   float X[3][4] = { 12.34, 34.56, 56.78, 78.90,
                     23.45, 45.67, 67.89, 89.01,
                     34.56, 56.78, 78.90, 90.12};

/*Menampilkan dalam bentuk matrik*/
        for(I=0;I<3;I++){
                for(J=0;J<3;J++)
                printf("%8.2f", X[I][J]);
                printf("\n");
                }
                printf("\n");      }
```

Output :

```
12.34   34.56   56.78   78.90
23.45   45.67   67.89   89.01
34.56   56.78   78.90   90.12
```

Larik String

Hubungan antara nilai larik string dengan nilai larik karakter

String	Character
Nilai string tunggal	Larik karakter dimensi satu
Larik string dimensi satu	Larik karakter dimensi dua
Larik string dimensi dua	Larik karakter dimensi tiga
Larik string dimensi N	Larik karakter dimensi N + 1

Contoh 1 :

```
#include <stdio.h>
main(){
        int I,J;
        char Hari[7][10] = {"Minggu", "Senin", "Selasa", "Rabu", "Kamis", "Jum'at", "Sabtu" };
        for(I=0;I<7;I++)
        {       for(J=0;J<10;J++)
                printf("%c", Hari[I][J]);
                printf("\n"); }
        }
```

Output :

```
Minggu
Senin
Selasa
Rabu
Kamis
Jum'at
Sabtu
```

Contoh 2 :

```
#include <stdio.h>
main(){
        int I;
        char Hari[7][10] = {"Minggu", "Senin", "Selasa", "Rabu", "Kamis", "Jum'at", "Sabtu" };
        for(I=0;I<7;I++)
        printf("%s \n", Hari[I]);
        }
```

Output :

```
Minggu
Senin
Selasa
Rabu
Kamis
Jum'at
Sabtu
```

Contoh 3 :

```
#include <stdio.h>
main(){
        int I,J;
        char Nama[5][3][10] = {"Adit", "Bayu", "Coki", "Dhani", "Erwin", "Fenty", "Gunarto", "Henry",
        "Ibrahim", "Joko", "Kemal", "Lukman", "Meny", "Nony", "Onie" };
        for(I=0;I<5;I++)
        {       printf("Nama-nama dikelas %1d adalah       : \n", I+1);
                for(J=0;J<3;J++)
                        printf("%s \n", Nama[I][J]);
                        }       }
```

Output :

```
Nama-nama dikelas 1 adalah  :
Adit
Bayu
Coki
Nama-nama dikelas 2 adalah  :
Dhani
Erwin
Fenty
Nama-nama dikelas 3 adalah  :
Gunarto
Henry
Ibrahim
Nama-nama dikelas 4 adalah  :
```

Joko
Kemal
Lukman
Nama-nama dikelas 5 adalah :
Meny
Nony
Onie

POINTER

Suatu pointer (variabel penunjuk) adalah suatu variabel yang berisi dengan alamat lokasi suatu memori tertentu. Bahasa C menyediakan 2 buah operator untuk operasi pointer yaitu operator '*' dan operator '&'.

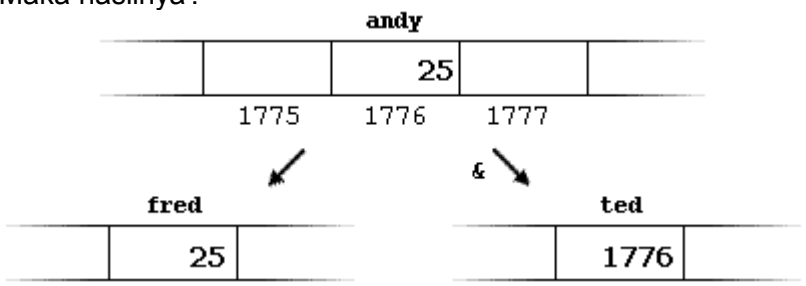
Operator Alamat (Address operator (&))
Pada saat pendeklarasian variable, user tidak diharuskan menentukan lokasi sesungguhnya pada memory, hal ini akan dilakukan secara otomatis oleh kompilerdan operating sysem pada saat run-time. Jika ingin mengetahui dimana suatu variable akan disimpan, dapat dilakukan dengan memberikan tanda *ampersand* (&) didepan variable , yang berarti *"address of"*.
Contoh:

```
ted = &andy;
```

Akan memberikan variable ted alamat dari variable andy, karena variable andy diberi awalan karakter *ampersand* (&), maka yang menjadi pokok disini adalah alamat dalam memory, bukan isi variable.
Misalkan andy diletakkan pada alamat 1776 kemudian dituliskan instruksi sbb :

```
andy = 25;
fred = andy;
ted = &andy;
```

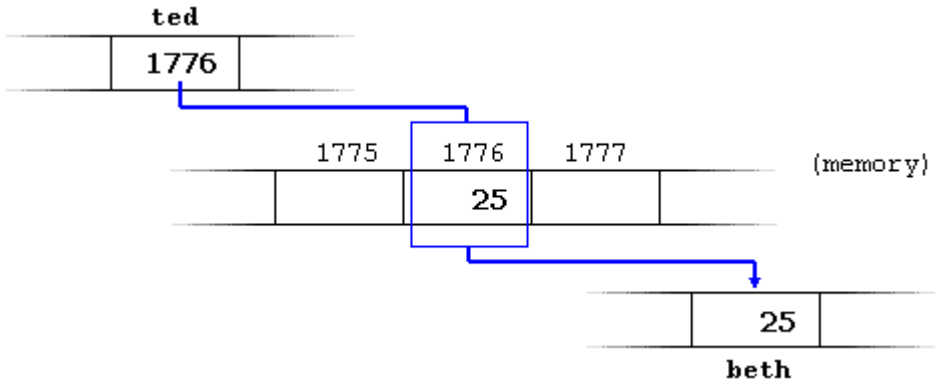
Maka hasilnya :



Operator Reference (*)
Dengan menggunakan pointer, kita dapat mengakses nilai yang tersimpan secara langsung dengan memberikan awalan operator *asterisk* (*) pada identifier pointer, yang berarti *"value pointed by"*.
Contoh:

```
beth = *ted;
```

(dapat dikatakan:"beth sama dengan nilai yang ditunjuk oleh ted") beth = 25, karena ted dialamat 1776, dan nilai yang berada pada alamat 1776 adalah 25.



Ekspresi dibawah ini semuanya benar, perhatikan :

andy == 25
&andy == 1776
ted == 1776
*ted == 25

Ekspresi pertama merupakan *assignation* bahwa andy=25;. Kedua, menggunakan operator alamat (address/derefence operator (&)), sehingga akan mengembalikan alamat dari variabel andy. Ketiga bernilai benar karena *assignation* untuk ted adalah ted = &andy;. Keempat menggunakan reference operator (*) yang berarti nilai yang ada pada alamat yang ditunjuk oleh ted, yaitu 25. Maka ekspresi dibawah ini pun akan bernilai benar :

*ted == andy

Contoh:

```
#include <stdio.h>
int main (){
  int value1 = 5, value2 = 15;
  int * mypointer;

  mypointer = &value1;
  *mypointer = 10;
  mypointer = &value2;
  *mypointer = 20;
  printf (“value1== %d ", value1);
  printf (“\Nvalue2== %d “, value2);
  return 0;
}
```

Output :
value1==10 / value2==20

Deklarasi Pointer

Variabel pointer dideklarasikan dengan nama variabelnya ditulis dengan diawali karakter asterik. Bentuk umum :

Tipe-data *nama-variabel-pointer;

Variabel pointer yang dideklarasikan dapat juga langsung diberikan nilai awal. Variabel pointer harus dideklarasikan dengan tipe yang sesuai dengan tipe data di memori yang ditunjuknya. Misalnya data yang ada dimemori bertipe float(4 byte) dan variabel pointer yang menunjukkan kealamat ini adalah bertipe int(2 byte), maka hanya 2 byte pertama saja yang akan diambil dengan hasil yang tidak sesuai dengan yang diharapkan.

Array dan Pointer

Identifier suatu array equivalen dengan alamat dari elemen pertama, pointer equivalen dengan alamat elemen pertama yang ditunjuk. Perhatikan deklarasi berikut :

```
int numbers [20];
int * p;
```

maka deklarasi dibawah ini juga benar :

```
p = numbers;
```

p dan numbers equivalen, dan memiliki sifat (*properties*) yang sama. Perbedaannya, user dapat menentukan nilai lain untuk pointer p dimana numbers akan selalu menunjuk nilai yang sama seperti yang telah didefinisikan. p, merupakan *variable pointer*, numbers adalah *constant pointer*. Karena itu walaupun instruksi diatas benar, tetapi tidak untuk instruksi dibawah ini :

```
numbers = p;
```

karena numbers adalah array (constant pointer), dan tidak ada nilai yang dapat diberikan untuk identifier konstant (*constant identifiers*).

Contoh:

```
#include <stdio.h>
int main (){
  int numbers[5];
  int * p;
  p = numbers;  *p = 10;
  p++;  *p = 20;
  p = &numbers[2];  *p = 30;
  p = numbers + 3;  *p = 40;
  p = numbers;  *(p+4) = 50;
  for (int n=0; n<5; n++)
        printf ("%d, ",  numbers[n]);
  return 0;}
```

Output :

```
10, 20, 30, 40, 50,
```