# MODUL 1 PROGRAMMING

## 1. Instalasi Bahasa C/C++

- For Windows : https://www.rose-hulman.edu/class/csse/resources/MinGW/installation.htm
- For Linux : udah ada
- For MacOS : mohon cari sendiri bila belum ada

Mengapa bahasa C yang dipelajari ?

Kelebihan Bahasa C : Kode yang compact, efisien , masih readable.

Kekurangan Bahasa C : Masih kurang readable dibanding Bahasa tingkat tinggi lain.

Bahasa ini cukup banyak digunakan oleh tim-tim yang ada di URO. Walaupun lebih tepatnya bahasa C++ yang digunakan karena pola pikir pemrograman object oriented, bahasa C merupakan sebuah dasar yang baik untuk mempelajari algoritma dasar dan sintaks-sintaks yang akan sering digunakan kedepannya.

## 2. Variabel dan Type Data

| Data Type | Memory (bytes) | Range | Data Specifier |
|---|---|---|---|
| int | 4 | -2,147,483,648 to 2,147,483,647 | %d |
| char | 1 | | %c |
| float | 4 | | %f |
| double | 8 | | %lf |
| short int | 2 | -32,768 to 32,767 | %hd |
| unsigned int | 4 | 0 to 4,294,967,295 | %u |
| long int | 4 | -2,147,483,648 to 2,147,483,647 | %li |
| long long int | 8 | -(2^63) to (2^63)-1 | %lli |
| unsigned long int | 4 | 0 to 4,294,967,295 | %lu |
| unsigned long long int | 8 | 0 to 18,446,744,073,709,551,615 | %llu |

# 3. Input/Output

Untuk C (dan C++) dengan menggunakan reference “stdio.h”, format Input dan Output harus didkelarasikan terlebih dahulu. Format input yang diajarkan pada modul ini hanya menggunakan header “stdio.h”. Untuk format yang lain dapat dillihat lebih lanjut di sumber lain.

Format Input :

**scanf(“<format>”, <list-nama>);**

Penggunaan & digunakan untuk mendapatkan alamat dari variable. (“&x” adalah alamat variable “x”) Pada contoh, dapat dilihat bahwa nilai yang dimasukkan oleh pengguna akan disimpan oleh address tersebut.
Contoh penggunaan:

- scanf("%d",&X); /*x integer*/
- scanf("%d %d", &X, &Y);
- scanf("%f",&F); /*F real*/
- scanf("%s",s); /*s string*/

Format Output :

**printf(“<format>”, <list-nama>);**

Untuk format dapat dilihat pada tabel diatas
Output mengirimkan output ke layar dalam bentuk string. Untuk memasukkan variable, kita harus menuliskan format output ke dalam string.
Contoh penggunaan :

- printf("Nilai dari : %d dari kelas %d",X,Y); /* x integer */
- printf("%d %d", X, Y);
- printf("Contoh output");
- printf("Namaku: %s", nama);
- printf("%f",F); /* F real */
- printf("%c",CC); /* CC char */

Contoh Program input/output di C :

```c
#include <stdio.h>

int main(){
    int x; //Variabel build di c, type data harus di deklarasikan
    x = 5; //Variabel assignment di c
    /* Kedua proses di atas sama dengan int x = 5 */
    int y;
```

```
    scanf("%d", &y);//input di c
    printf("y = %d", y);//output yang keluar adalah y = 5
    return 0;
}
```

# 4. If-Else Statement

Penggunaan If-Else :

```
if (kondisi)
{
      // Aksi
} else if (kondisi ){
      //Aksi
} else {
      // Aksi
}
```

Contoh kode menggunakan If-Else :

```
#include <stdio.h>

int main(){

    int nomor;

    scanf("%d", nomor);

    if (nomor < 5){//Kondisi 1

        printf("nomor kecil dari 5");

    }

    else { // Kondisi 2 berarti jika nomor >= 5 masuk ke kondisi
satu tidak terpenuhi

        printf("nomor besar dari 5");

    }

}
```

Penggunaan Switch :

```
Switch (nama) {
      case const-exp-1: aksi-1;
            break;
      case const-exp-2: aksi-2;
            break;
      …
      Default: aksi-else;
            break;
};
```

Contoh Program menggunakan switch :

```
#include <stdio.h>

int main() {

  int day;

  scanf("%d", &day);

  switch (day) { // Kondisi berdasarkan variabel day

  case 1: // sama dengan if (day == 1)

    printf("Monday");

    break;

  case 2:

    printf("Tuesday");

    break;

  case 3:

    printf("Wednesday");

    break;

  case 4:

    printf("Thursday");

    break;
```

```
  case 5:
    printf("Friday");
    break;
  case 6:
    printf("Saturday");
   break;
  case 7:
    printf("Sunday");
    break;
  default :
    printf("No day");
  }
  return 0;
}
```

# 5. Operators

a. *Arithmetic Operators*

| Syntax | Keterangan |
|---|---|
| + (Addition) | Operasi penjumlahan<br>Contoh : 13 + 14 = 27 |
| - (Subtraction) | Operasi pengurangan<br>Contoh : 13 - 14 = -1 |
| * (Multiplication) | Operasi perkalian<br>Contoh : 5 * 3 = 15 |
| / (Division) | Operasi pembagian<br>Contoh :<br>- Pada tipe data float atau double :<br>- Pada tipe data integer : |
| % (Modulus) | Mengembalikan sisa pembagian (*remainder*) antara 2 integer. |

| | |
|---|---|
| | Catatan : hanya bisa digunakan untuk integer |

*b. Relational and Logical Operators*

*1) Relational*

| Syntax | Keterangan |
|---|---|
| <= (Less than or equal to) | a <= b mengembalikan 1 atau *true* jika a kurang dari atau sama dengan b.<br>Contoh : 6 <= 4 mengambalikan 0 (false) |
| >= (Greater than or equal to) | a >= b mengembalikan 1 atau *true* jika a lebih dari atau sama dengan b.<br>Contoh : 6 >= 4 mengambalikan 1 (true) |
| > (Greater than) | a > b mengembalikan 1 atau *true* jika a lebih besar dari b.<br>Contoh : 6 > 4 mengambalikan 1 (true) |
| < (Less than) | a < b mengembalikan 1 atau *true* jika a lebih kecil dari b.<br>Contoh : 6 < 4 mengambalikan 0 (false) |
| == (Equal to) | a == b mengembalikan 1 atau *true* jika a sama dengan b.<br>Contoh : 6 == 4 mengambalikan 0 (false) |
| != (Not equal to) | a == b mengembalikan 1 atau *true* jika a tidak sama dengan b.<br>Contoh : 6 != 4 mengambalikan 1 (true) |

*2) Logical*

| Syntax | Keterangan |
|---|---|
| && (Logical AND) | a && b mengembalikan 1 atau *true* jika a dan b bernilai benar keduanya. |

| | |
|---|---|
| \|\| (Logical OR) | a \|\| b mengembalikan 1 atau *true* jika a atau b bernilai benar, atau keduanya bernilai benar. |
| ! (Logical NOT) | !a mengembalikan *true* jika a bernilai salah (*false*) atau a bernilai 0. |

c. *Increment and Decrement Operators*

| Syntax | Keterangan |
|---|---|
| ++ (Increment) | Menambah sebuah variabel atau identifier dengan 1. Bisa digunakan sebagai prefix atau postfix.<br>Contoh :<br>int x = 1;<br>++x; // x menjadi 2<br>X++; // x menjadi 3 |
| -- (Decrement) | Mengurangi sebuah variabel atau identifier dengan 1. Bisa digunakan sebagai prefix atau postfix.<br>Contoh :<br>int x = 3;<br>--x; // x menjadi 2<br>X--; // x menjadi 1 |

Terdapat perbedaan penggunaan ketika kedua operator tersebut digunakan sebagai prefix dengan postfix. Ketika kedua operator tersebut digunakan sebagai prefix, maka nilai pada identifier atau variabel akan dievaluasi terlebih dahulu (baik *increment* maupun *decrement*), baru nilai barunya digunakan dalam sebuah ekspresi misalnya. Untuk postfix, nilai lama dari identifier akan digunakan dalam sebuah ekspresi misalnya, baru kemudian nilai identifier baru dihitung (*increment* atau *decrement*).

Contoh Penggunaan :

```
int x = 3;
printf("%d",++x) ;        // print 4
printf("%d",x++) ;        // print 4
printf("%d",x);           //print 5
```

d. *Bitwise Operators*

Digunakan untuk mengoperasikan representasi biner dari dua operand.

| Syntax | Keterangan |
|---|---|
| & (Bitwise AND) | Menghasilkan bit 1 jika dua bit yang berkorespondensi pada kedua operand bernilai 1.<br>Contoh : 01010010 & 00110010 = 00010010 |
| \| (Bitwise OR) | Menghasilkan bit 1 jika dua bit yang berkorespondensi pada kedua operand bernilai 1 atau hanya salah satu yang bernilai 1.<br>Contoh : 01010010 & 00110010 = 01110010 |
| ^ (Bitwise XOR) | Menghasilkan bit 1 jika dua bit yang berkorespondensi pada kedua operand salah satunya bernilai 1, namun tidak keduanya.<br>Contoh : 01010010 & 00110010 = 01100000 |
| << (Shift LEFT) | (<< a) menggeser tiap bit pada a ke kiri, misalnya sebanyak x, dengan menghilangkan x bit paling kiri dan menambah bit 0 sebanyak x pada posisi kanan.<br>Contoh :<br>int x = 2;<br>x = x << 1;<br><br>Maka nilai x menjadi 4 |
| >> (Shift RIGHT) | (>> a) menggeser tiap bit pada a ke kanan, misalnya sebanyak x, dengan menghilangkan x bit paling kanan. Kemudian, untuk unsigned menambah bit 0 sebanyak x pada posisi kiri, namun untuk tipe data signed menambah bit 1 sebanyak x pada posisi kiri jika bit paling kiri nya sama dengan 1 (berlaku pada beberapa mesin).<br>Contoh :<br>int x = 2;<br>x = x >> 1;<br><br>Maka nilai x menjadi 1 |
| ~ (One's complement) | Sebuah *unary operator* (~a) yang mengubah semua bit 1 pada a menjadi 0 dan semua bit 0 pada a menjadi 1. |

*e. Assignment Operators*

*Assignment Operators* merupakan operator dalam bentuk $expr_1$ op= $expr_2$ yang ekuivalen dengan $expr_1$ = $expr_1$ op $expr_2$, dengan op merupakan salah satu dari +, -, *, /, %, <<, >>, &, ^, |.

# 6. Loops

a. While Loop

*While Loop* merupakan loop dengan bentuk umum :

```
while (expr1) {
        statements
}
```

b. For Loop

*For Loop* merupakan loop dengan bentuk umum :

```
for (expr1; expr2; expr3) {
        statements
}
```

Yang ekuivalen dengan :

```
expr1;
while (expr2) {
   statements
   expr3;
}
```

c. Do-While Loop

*Do-While Loop* merupakan loop dengan bentuk umum :

```
do {
        statements
} while(expr1);
```

# 7. Fungsi

| **NOTASI ALGORITMIK** |
| --- |
| function NAMAF (param1: type1, param2: type2, …) → type-hasil<br>{ Spesifikasi fungsi } |

**KAMUS LOKAL**
{ Semua nama yang dipakai dalam algoritma dari fungsi }

**ALGORITMA**
{ Deretan fungsi algoritmik: pemberian harga, input, output, analisis kasus, pengulangan }

{ Pengiriman harga di akhir fungsi, harus sesuai dengan type-hasil }

→ hasil

**BAHASA C**

type-hasil NAMAF (type1 param1, type2 param2, ...)
/* Spesifikasi fungsi */
{

/* KAMUS LOKAL */
/* Semua nama yang dipakai dalam algoritma dari fungsi */

/* ALGORITMA */
/* Deretan fungsi algoritmik: pemberian harga, input, output, analisis kasus, pengulangan */

/* Pengiriman harga di akhir fungsi, harus sesuai dengan type-hasil */

return(hasil);
}

**CONTOH FUNGSI DALAM BAHASA C**

float luasSegitiga (float alas, float tinggi)
/* Fungsi untuk menghitung luas segitiga */
{
/* KAMUS LOKAL */

/* ALGORITMA */

```
return(0.5 * alas * tinggi);
}
```

## Pemanggilan fungsi

**NOTASI ALGORITMIK**

```
Program SUATU_PROGRAM
{ Spesifikasi: Input, Proses, Output }
```

**KAMUS**

```
{semua nama yang dipakai dalam algoritma}

{Prototype fungsi}
function NAMAF ([list nama:type input]) → type-hasil
{Spesifikasi fungsi}
```

**ALGORITMA**

```
{Deretan instruksi pemberian harga, input, output, analisa kasus, pengulangan yang memakai fungsi}

{Harga yang dihasilkan fungsi juga dapat dipakai dalam ekspresi}

nama ← NAMAF ([list parameter aktual])

output (NAMAF ([list parameter aktual]))

{Body/Realisasi Fungsi diletakkan setelah program utama}
```

**PEMANGGILAN FUNGSI DALAM BAHASA C**

```
/* Program SUATU_PROGRAM */
/* Spesifikasi: Input, Proses, Output */

/* Prototype Fungsi */
```

```
type-hasil NAMAF ([list <type nama> input]);
/* Spesifikasi Fungsi */

int main () {

/* KAMUS */
/* Semua nama yang dipakai dalam algoritma */

/* ALGORITMA */
/* Deretan instruksi pemberian harga, input, output, analisis
kasus, pengulangan yang memakai fungsi */

/* Harga yang dihasilkan fungsi juga dapat dipakai dalam
ekspresi */

nama = NAMAF ([list parameter aktual]);
printf ("[format]", NAMAF ([list parameter aktual]));

/* Harga yang dihasilkan fungsi juga dapat dipakai dalam
ekspresi */

return 0;
}

/* Body/Realisasi Fungsi diletakkan setalah program utama */
```

**CONTOH PEMANGGILAN FUNGSI DALAM BAHASA C**

```
/* Program LUAS_SEGITIGA */
/* Program untuk menghitung luas segitiga */

/* Prototype Fungsi */
float luasSegitiga (float alas, float tinggi)
/* Fungsi untuk menghitung luas segitiga */
{
/* KAMUS LOKAL */

/* ALGORITMA */
```

```
return(0.5 * alas * tinggi);
}

int main () {

/* KAMUS */
float alas, tinggi, luas;

/* ALGORITMA */
/* input */
scanf("%f", &alas);
scanf("%f", &tinggi);

/* pemanggilan fungsi */
luas = luasSegitiga(alas,tinggi);

/* output */
printf("Luas : %.2f", luas);

return 0;
}
```

# 8. Prosedur

| **NOTASI ALGORITMIK** |
| --- |
| procedure NAMAP (input nama1: type1, input/output nama2: type2, output nama3: type3)<br>{ Spesifikasi, Initial State, Final State } |
| **KAMUS LOKAL**<br>{ Semua nama yang dipakai dalam BADAN PROSEDUR } |

**ALGORITMA**
{ BADAN PROSEDUR }
{ Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan atau prosedur }

{ Pengiriman harga di akhir fungsi, harus sesuai dengan type-hasil }

**BAHASA C**

```
void NAMAP (type1 nama1, type2 *nama2, type3 *nama3)
/* Spesifikasi, Initial State, Final State */

{
/* KAMUS LOKAL */
/* Semua nama yang dipakai dalam BADAN PROSEDUR */

/* ALGORITMA */
/* Deretan instruksi pemberian harga, input, output,
analisis kasus, pengulangan atau prosedur */
}
```

**CONTOH PROSEDUR DALAM BAHASA C**

```
void showMenu(int input){
/* Prosedur untuk menampilka daftar menu */
   printf("\nMenu\n");
   printf("====\n");
   printf("1. Add items\n");
   printf("2. See items\n");
   printf("3. Search items\n");
   printf("4.Exit\n");
}
```

## Pemanggilan prosedur

**NOTASI ALGORITMIK**

Program SUATU_PROGRAM
{ Spesifikasi: Input, Proses, Output }

**KAMUS**
{semua nama yang dipakai dalam algoritma}

{Prototype prosedur}
procedure NAMAP (input nama1: type1, input/output nama2: type2, output nama3: type3)
{Spesifikasi prosedur, initial state, final state}

**ALGORITMA**
{Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan}
NAMAP(paramaktual1, paramaktual2, paramaktual3)

{Body/Realisasi Prosedur diletakkan setelah program utama}

**PEMANGGILAN PROSEDUR DALAM BAHASA C**

```
/* Program SUATU_PROGRAM */
/* Spesifikasi: Input, Proses, Output */

/* Prototype Fungsi */
void NAMAP (type1 nama1, type2 *nama2, type3 *nama3);
/* Spesifikasi prosedur, initial state, final state */
int main () {

/* KAMUS */
/* semua nama yang dipakai dalam algoritma */

/* ALGORITMA */
/* Deretan instruksi pemberian harga, input, output,
analisis kasus, pengulangan */

NAMAP(paramaktual1, &paramaktual2, &paramaktual3);

return 0;
}
```

```
/* Body/Realisasi Fungsi diletakkan setalah program utama */
```

**CONTOH PROGRAM C BESERTA PEMANGGILAN PROSEDUR DALAM BAHASA C**

```
/* Program OLAH_ITEM */
/* Program untuk mengolah item */

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

/* Prototype Prosedur */
void showMenu(int input){
   printf("\nMenu\n");
   printf("====\n");
   printf("1. Add items\n");
   printf("2. See items\n");
   printf("3. Search items\n");
   printf("4.Exit\n");
   printf("Pilih Menu[1..%d]: ",input);
}

int main (){
   int input;
   int menu;

   typedef struct{
      char* name;
      int qty;
   } Item;

   printf("Input storage[1..10]: ");
   scanf("%d",&input);
   while ((input<1)||(input>10)){
      printf("wrong input\n");
      printf("Input storage[1..10]: ");
      scanf("%d",&input);
   }
```

```
Item listItem[input];
int i=0;

showMenu(input);
scanf("%d",&menu);

while (menu!=4){
   if (menu==1){
      if (i<input){
         char* name = (char*) malloc (100 * sizeof(char));
         printf("name: ");
         scanf("%s",name);
         int qty;
         printf("quantity: ");
         scanf("%d",&qty);
         Item item;
         item.name=name;
         item.qty=qty;

         listItem[i] = item;
         i++;
      }
      else {
         printf("Storage is full\n");
      }
      showMenu(input);
      scanf("%d",&menu);
   }
   else if (menu==2){
      if(i==0) {
         printf("Storage is Empty");
      }
      else {
         printf("No. Name       Quantity\n");
         for (int a=1;a<=i;a++){
            printf("%d. %s      %d\n",a,listItem[a-1].name,listItem[a-1].qty);
         }
      }
      showMenu(input);
```

```
            scanf("%d",&menu);
        }
        else if (menu==3){
            int a;
            printf("Input number of items that you want to see [1..%d]: ",input);
            scanf("%d",&a);
            while ((a<1)||(a>input)){
                printf("wrong input\n");
                 printf("Input number of items that you want to see [1..%d]: ",input);
                scanf("%d",&a);
            }
            if (a<=input){
                printf("No. Name       Quantity\n");
                printf("%d. %s      %d\n",a,listItem[a-1].name,listItem[a-1].qty);
            }
            else{
                printf("No item with this number\n");
            }
            showMenu(input);
            scanf("%d",&menu);
        }
    }

    return 0;
}
```

1

# MODUL 1 PROGRAMMING

[1]Seluruh penjelasan tersebut, merupakan sintaks-sintaks dan algoritma paling dasar yang diperlukan untuk pemrograman menggunakan bahasa C. Beberapa hal memang mungkin perlu diingat-ingat agar lebih leluasa saat akan melakukan pemrograman kedepannya. Namun pahami bahwa yang terpenting bukanlah memahami sintaks yang ada, namun untuk memahami bagaimana pola pikir yang tepat untuk melakukan pemrograman.

# 1. Instalasi Bahasa C/C++

- For Windows : https://www.rose-hulman.edu/class/csse/resources/MinGW/installation.htm
- For Linux : udah ada
- For MacOS : mohon cari sendiri bila belum ada

Mengapa bahasa C yang dipelajari ?

Kelebihan Bahasa C : Kode yang compact, efisien , masih readable.

Kekurangan Bahasa C : Masih kurang readable dibanding Bahasa tingkat tinggi lain.

Bahasa ini cukup banyak digunakan oleh tim-tim yang ada di URO. Walaupun lebih tepatnya bahasa C++ yang digunakan karena pola pikir pemrograman object oriented, bahasa C merupakan sebuah dasar yang baik untuk mempelajari algoritma dasar dan sintaks-sintaks yang akan sering digunakan kedepannya.

# 2. Variabel dan Type Data

| Data Type | Memory (bytes) | Range | Data Specifier |
|---|---|---|---|
| int | 4 | -2,147,483,648 to 2,147,483,647 | %d |
| char | 1 | | %c |
| float | 4 | | %f |
| double | 8 | | %lf |
| short int | 2 | -32,768 to 32,767 | %hd |
| unsigned int | 4 | 0 to 4,294,967,295 | %u |
| long int | 4 | -2,147,483,648 to 2,147,483,647 | %li |
| long long int | 8 | -(2^63) to (2^63)-1 | %lli |
| unsigned long int | 4 | 0 to 4,294,967,295 | %lu |
| unsigned long long int | 8 | 0 to 18,446,744,073,709,551,615 | %llu |

# 3. Input/Output

Untuk C (dan C++) dengan menggunakan reference “stdio.h”, format Input dan Output harus didkelarasikan terlebih dahulu. Format input yang diajarkan pada modul ini hanya menggunakan header “stdio.h”. Untuk format yang lain dapat dillihat lebih lanjut di sumber lain.

Format Input :

**scanf(“<format>”, <list-nama>);**

Penggunaan & digunakan untuk mendapatkan alamat dari variable. (“&x” adalah alamat variable “x”) Pada contoh, dapat dilihat bahwa nilai yang dimasukkan oleh pengguna akan disimpan oleh address tersebut.
Contoh penggunaan:

- scanf("%d",&X); /*x integer*/
- scanf("%d %d", &X, &Y);
- scanf("%f",&F); /*F real*/
- scanf("%s",s); /*s string*/

Format Output :

**printf(“<format>”, <list-nama>);**

Untuk format dapat dilihat pada tabel diatas
Output mengirimkan output ke layar dalam bentuk string. Untuk memasukkan variable, kita harus menuliskan format output ke dalam string.
Contoh penggunaan :

- printf("Nilai dari : %d dari kelas %d",X,Y); /* x integer */
- printf("%d %d", X, Y);
- printf("Contoh output");
- printf("Namaku: %s", nama);
- printf("%f",F); /* F real */
- printf("%c",CC); /* CC char */

Contoh Program input/output di C :

```c
#include <stdio.h>

int main(){
    int x; //Variabel build di c, type data harus di deklarasikan
    x = 5; //Variabel assignment di c
    /* Kedua proses di atas sama dengan int x = 5 */
    int y;
```

```
    scanf("%d", &y);//input di c
    printf("y = %d", y);//output yang keluar adalah y = 5
    return 0;
}
```

# 4. If-Else Statement

Penggunaan If-Else :

```
if (kondisi)
{
      // Aksi
} else if (kondisi ){
      //Aksi
} else {
      // Aksi
}
```

Contoh kode menggunakan If-Else :

```
#include <stdio.h>

int main(){

    int nomor;

    scanf("%d", nomor);

    if (nomor < 5){//Kondisi 1

        printf("nomor kecil dari 5");

    }

    else { // Kondisi 2 berarti jika nomor >= 5 masuk ke kondisi
satu tidak terpenuhi

        printf("nomor besar dari 5");

    }

}
```

Penggunaan Switch :

```
Switch (nama) {
      case const-exp-1: aksi-1;
            break;
      case const-exp-2: aksi-2;
            break;
      …
      Default: aksi-else;
            break;
};
```

Contoh Program menggunakan switch :

```
#include <stdio.h>

int main() {

  int day;

  scanf("%d", &day);

  switch (day) { // Kondisi berdasarkan variabel day

  case 1: // sama dengan if (day == 1)

    printf("Monday");

    break;

  case 2:

    printf("Tuesday");

    break;

  case 3:

    printf("Wednesday");

    break;

  case 4:

    printf("Thursday");

    break;
```

```
  case 5:
    printf("Friday");
    break;
  case 6:
    printf("Saturday");
   break;
  case 7:
    printf("Sunday");
    break;
  default :
    printf("No day");
  }
  return 0;
}
```

# 5. Operators

a. *Arithmetic Operators*

| Syntax | Keterangan |
|---|---|
| + (Addition) | Operasi penjumlahan<br>Contoh : 13 + 14 = 27 |
| - (Subtraction) | Operasi pengurangan<br>Contoh : 13 - 14 = -1 |
| * (Multiplication) | Operasi perkalian<br>Contoh : 5 * 3 = 15 |
| / (Division) | Operasi pembagian<br>Contoh :<br>- Pada tipe data float atau double :<br>- Pada tipe data integer : |
| % (Modulus) | Mengembalikan sisa pembagian (*remainder*) antara 2 integer. |

| | |
|---|---|
| | Catatan : hanya bisa digunakan untuk integer |

*b. Relational and Logical Operators*

*1) Relational*

| Syntax | Keterangan |
|---|---|
| <= (Less than or equal to) | a <= b mengembalikan 1 atau *true* jika a kurang dari atau sama dengan b.<br>Contoh : 6 <= 4 mengambalikan 0 (false) |
| >= (Greater than or equal to) | a >= b mengembalikan 1 atau *true* jika a lebih dari atau sama dengan b.<br>Contoh : 6 >= 4 mengambalikan 1 (true) |
| > (Greater than) | a > b mengembalikan 1 atau *true* jika a lebih besar dari b.<br>Contoh : 6 > 4 mengambalikan 1 (true) |
| < (Less than) | a < b mengembalikan 1 atau *true* jika a lebih kecil dari b.<br>Contoh : 6 < 4 mengambalikan 0 (false) |
| == (Equal to) | a == b mengembalikan 1 atau *true* jika a sama dengan b.<br>Contoh : 6 == 4 mengambalikan 0 (false) |
| != (Not equal to) | a == b mengembalikan 1 atau *true* jika a tidak sama dengan b.<br>Contoh : 6 != 4 mengambalikan 1 (true) |

*2) Logical*

| Syntax | Keterangan |
|---|---|
| && (Logical AND) | a && b mengembalikan 1 atau *true* jika a dan b bernilai benar keduanya. |

| | |
|---|---|
| \|\| (Logical OR) | a \|\| b mengembalikan 1 atau *true* jika a atau b bernilai benar, atau keduanya bernilai benar. |
| ! (Logical NOT) | !a mengembalikan *true* jika a bernilai salah (*false*) atau a bernilai 0. |

c. *Increment and Decrement Operators*

| Syntax | Keterangan |
|---|---|
| ++ (Increment) | Menambah sebuah variabel atau identifier dengan 1. Bisa digunakan sebagai prefix atau postfix.<br>Contoh :<br>int x = 1;<br>++x; // x menjadi 2<br>X++; // x menjadi 3 |
| -- (Decrement) | Mengurangi sebuah variabel atau identifier dengan 1. Bisa digunakan sebagai prefix atau postfix.<br>Contoh :<br>int x = 3;<br>--x; // x menjadi 2<br>X--; // x menjadi 1 |

Terdapat perbedaan penggunaan ketika kedua operator tersebut digunakan sebagai prefix dengan postfix. Ketika kedua operator tersebut digunakan sebagai prefix, maka nilai pada identifier atau variabel akan dievaluasi terlebih dahulu (baik *increment* maupun *decrement*), baru nilai barunya digunakan dalam sebuah ekspresi misalnya. Untuk postfix, nilai lama dari identifier akan digunakan dalam sebuah ekspresi misalnya, baru kemudian nilai identifier baru dihitung (*increment* atau *decrement*).

Contoh Penggunaan :

```
int x = 3;
printf("%d",++x) ;        // print 4
printf("%d",x++) ;        // print 4
printf("%d",x);           //print 5
```

d. *Bitwise Operators*

Digunakan untuk mengoperasikan representasi biner dari dua operand.

| Syntax | Keterangan |
|---|---|
| & (Bitwise AND) | Menghasilkan bit 1 jika dua bit yang berkorespondensi pada kedua operand bernilai 1.<br>Contoh : 01010010 & 00110010 = 00010010 |
| \| (Bitwise OR) | Menghasilkan bit 1 jika dua bit yang berkorespondensi pada kedua operand bernilai 1 atau hanya salah satu yang bernilai 1.<br>Contoh : 01010010 & 00110010 = 01110010 |
| ^ (Bitwise XOR) | Menghasilkan bit 1 jika dua bit yang berkorespondensi pada kedua operand salah satunya bernilai 1, namun tidak keduanya.<br>Contoh : 01010010 & 00110010 = 01100000 |
| << (Shift LEFT) | (<< a) menggeser tiap bit pada a ke kiri, misalnya sebanyak x, dengan menghilangkan x bit paling kiri dan menambah bit 0 sebanyak x pada posisi kanan.<br>Contoh :<br>int x = 2;<br>x = x << 1;<br><br>Maka nilai x menjadi 4 |
| >> (Shift RIGHT) | (>> a) menggeser tiap bit pada a ke kanan, misalnya sebanyak x, dengan menghilangkan x bit paling kanan. Kemudian, untuk unsigned menambah bit 0 sebanyak x pada posisi kiri, namun untuk tipe data signed menambah bit 1 sebanyak x pada posisi kiri jika bit paling kiri nya sama dengan 1 (berlaku pada beberapa mesin).<br>Contoh :<br>int x = 2;<br>x = x >> 1;<br><br>Maka nilai x menjadi 1 |
| ~ (One's complement) | Sebuah *unary operator* (~a) yang mengubah semua bit 1 pada a menjadi 0 dan semua bit 0 pada a menjadi 1. |

*e. Assignment Operators*

*Assignment Operators* merupakan operator dalam bentuk $expr_1$ op= $expr_2$ yang ekuivalen dengan $expr_1$ = $expr_1$ op $expr_2$, dengan op merupakan salah satu dari +, -, *, /, %, <<, >>, &, ^, |.

# 6. Loops

a. While Loop

*While Loop* merupakan loop dengan bentuk umum :

```
while (expr1) {
        statements
}
```

b. For Loop

*For Loop* merupakan loop dengan bentuk umum :

```
for (expr1; expr2; expr3) {
        statements
}
```

Yang ekuivalen dengan :

```
expr1;
while (expr2) {
   statements
   expr3;
}
```

c. Do-While Loop

*Do-While Loop* merupakan loop dengan bentuk umum :

```
do {
        statements
} while(expr1);
```

# 7. Fungsi

| **NOTASI ALGORITMIK** |
| --- |
| function NAMAF (param1: type1, param2: type2, …) → type-hasil<br>{ Spesifikasi fungsi } |

**KAMUS LOKAL**
{ Semua nama yang dipakai dalam algoritma dari fungsi }

**ALGORITMA**
{ Deretan fungsi algoritmik: pemberian harga, input, output, analisis kasus, pengulangan }

{ Pengiriman harga di akhir fungsi, harus sesuai dengan type-hasil }

→ hasil

**BAHASA C**

type-hasil NAMAF (type1 param1, type2 param2, ...)
/* Spesifikasi fungsi */
{

/* KAMUS LOKAL */
/* Semua nama yang dipakai dalam algoritma dari fungsi */

/* ALGORITMA */
/* Deretan fungsi algoritmik: pemberian harga, input, output, analisis kasus, pengulangan */

/* Pengiriman harga di akhir fungsi, harus sesuai dengan type-hasil */

return(hasil);
}

**CONTOH FUNGSI DALAM BAHASA C**

float luasSegitiga (float alas, float tinggi)
/* Fungsi untuk menghitung luas segitiga */
{
/* KAMUS LOKAL */

/* ALGORITMA */

```
return(0.5 * alas * tinggi);
}
```

## Pemanggilan fungsi

**NOTASI ALGORITMIK**

```
Program SUATU_PROGRAM
{ Spesifikasi: Input, Proses, Output }
```

**KAMUS**

```
{semua nama yang dipakai dalam algoritma}

{Prototype fungsi}
function NAMAF ([list nama:type input]) → type-hasil
{Spesifikasi fungsi}
```

**ALGORITMA**

```
{Deretan instruksi pemberian harga, input, output, analisa kasus, pengulangan yang memakai fungsi}

{Harga yang dihasilkan fungsi juga dapat dipakai dalam ekspresi}

nama ← NAMAF ([list parameter aktual])

output (NAMAF ([list parameter aktual]))

{Body/Realisasi Fungsi diletakkan setelah program utama}
```

**PEMANGGILAN FUNGSI DALAM BAHASA C**

```
/* Program SUATU_PROGRAM */
/* Spesifikasi: Input, Proses, Output */

/* Prototype Fungsi */
```

```
type-hasil NAMAF ([list <type nama> input]);
/* Spesifikasi Fungsi */

int main () {

/* KAMUS */
/* Semua nama yang dipakai dalam algoritma */

/* ALGORITMA */
/* Deretan instruksi pemberian harga, input, output, analisis
kasus, pengulangan yang memakai fungsi */

/* Harga yang dihasilkan fungsi juga dapat dipakai dalam
ekspresi */

nama = NAMAF ([list parameter aktual]);
printf ("[format]", NAMAF ([list parameter aktual]));

/* Harga yang dihasilkan fungsi juga dapat dipakai dalam
ekspresi */

return 0;
}

/* Body/Realisasi Fungsi diletakkan setalah program utama */
```

**CONTOH PEMANGGILAN FUNGSI DALAM BAHASA C**

```
/* Program LUAS_SEGITIGA */
/* Program untuk menghitung luas segitiga */

/* Prototype Fungsi */
float luasSegitiga (float alas, float tinggi)
/* Fungsi untuk menghitung luas segitiga */
{
/* KAMUS LOKAL */

/* ALGORITMA */
```

```
return(0.5 * alas * tinggi);
}

int main () {

/* KAMUS */
float alas, tinggi, luas;

/* ALGORITMA */
/* input */
scanf("%f", &alas);
scanf("%f", &tinggi);

/* pemanggilan fungsi */
luas = luasSegitiga(alas,tinggi);

/* output */
printf("Luas : %.2f", luas);

return 0;
}
```

# 8. Prosedur

| **NOTASI ALGORITMIK** |
| --- |
| procedure NAMAP (input nama1: type1, input/output nama2: type2, output nama3: type3)<br>{ Spesifikasi, Initial State, Final State } |
| **KAMUS LOKAL**<br>{ Semua nama yang dipakai dalam BADAN PROSEDUR } |

**ALGORITMA**
{ BADAN PROSEDUR }
{ Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan atau prosedur }

{ Pengiriman harga di akhir fungsi, harus sesuai dengan type-hasil }

**BAHASA C**

```
void NAMAP (type1 nama1, type2 *nama2, type3 *nama3)
/* Spesifikasi, Initial State, Final State */

{
/* KAMUS LOKAL */
/* Semua nama yang dipakai dalam BADAN PROSEDUR */

/* ALGORITMA */
/* Deretan instruksi pemberian harga, input, output,
analisis kasus, pengulangan atau prosedur */
}
```

**CONTOH PROSEDUR DALAM BAHASA C**

```
void showMenu(int input){
/* Prosedur untuk menampilka daftar menu */
   printf("\nMenu\n");
   printf("====\n");
   printf("1. Add items\n");
   printf("2. See items\n");
   printf("3. Search items\n");
   printf("4.Exit\n");
}
```

## Pemanggilan prosedur

**NOTASI ALGORITMIK**

Program SUATU_PROGRAM
{ Spesifikasi: Input, Proses, Output }

**KAMUS**
{semua nama yang dipakai dalam algoritma}

{Prototype prosedur}
procedure NAMAP (input nama1: type1, input/output nama2: type2, output nama3: type3)
{Spesifikasi prosedur, initial state, final state}

**ALGORITMA**
{Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan}
NAMAP(paramaktual1, paramaktual2, paramaktual3)

{Body/Realisasi Prosedur diletakkan setelah program utama}

**PEMANGGILAN PROSEDUR DALAM BAHASA C**

```
/* Program SUATU_PROGRAM */
/* Spesifikasi: Input, Proses, Output */

/* Prototype Fungsi */
void NAMAP (type1 nama1, type2 *nama2, type3 *nama3);
/* Spesifikasi prosedur, initial state, final state */
int main () {

/* KAMUS */
/* semua nama yang dipakai dalam algoritma */

/* ALGORITMA */
/* Deretan instruksi pemberian harga, input, output,
analisis kasus, pengulangan */

NAMAP(paramaktual1, &paramaktual2, &paramaktual3);

return 0;
}
```

```
/* Body/Realisasi Fungsi diletakkan setalah program utama */
```

**CONTOH PROGRAM C BESERTA PEMANGGILAN PROSEDUR DALAM BAHASA C**

```
/* Program OLAH_ITEM */
/* Program untuk mengolah item */

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

/* Prototype Prosedur */
void showMenu(int input){
   printf("\nMenu\n");
   printf("====\n");
   printf("1. Add items\n");
   printf("2. See items\n");
   printf("3. Search items\n");
   printf("4.Exit\n");
   printf("Pilih Menu[1..%d]: ",input);
}

int main (){
   int input;
   int menu;

   typedef struct{
      char* name;
      int qty;
   } Item;

   printf("Input storage[1..10]: ");
   scanf("%d",&input);
   while ((input<1)||(input>10)){
      printf("wrong input\n");
      printf("Input storage[1..10]: ");
      scanf("%d",&input);
   }
```

```
Item listItem[input];
int i=0;

showMenu(input);
scanf("%d",&menu);

while (menu!=4){
   if (menu==1){
      if (i<input){
         char* name = (char*) malloc (100 * sizeof(char));
         printf("name: ");
         scanf("%s",name);
         int qty;
         printf("quantity: ");
         scanf("%d",&qty);
         Item item;
         item.name=name;
         item.qty=qty;

         listItem[i] = item;
         i++;
      }
      else {
         printf("Storage is full\n");
      }
      showMenu(input);
      scanf("%d",&menu);
   }
   else if (menu==2){
      if(i==0) {
         printf("Storage is Empty");
      }
      else {
         printf("No. Name       Quantity\n");
         for (int a=1;a<=i;a++){
            printf("%d. %s      %d\n",a,listItem[a-1].name,listItem[a-1].qty);
         }
      }
      showMenu(input);
```

```
            scanf("%d",&menu);
        }
        else if (menu==3){
            int a;
            printf("Input number of items that you want to see [1..%d]: ",input);
            scanf("%d",&a);
            while ((a<1)||(a>input)){
                printf("wrong input\n");
                 printf("Input number of items that you want to see [1..%d]: ",input);
                scanf("%d",&a);
            }
            if (a<=input){
                printf("No. Name       Quantity\n");
                printf("%d. %s      %d\n",a,listItem[a-1].name,listItem[a-1].qty);
            }
            else{
                printf("No item with this number\n");
            }
            showMenu(input);
            scanf("%d",&menu);
        }
    }

    return 0;
}
```

2

[2]Seluruh penjelasan tersebut, merupakan sintaks-sintaks dan algoritma paling dasar yang diperlukan untuk pemrograman menggunakan bahasa C. Beberapa hal memang mungkin perlu diingat-ingat agar lebih leluasa saat akan melakukan pemrograman kedepannya. Namun pahami bahwa yang terpenting bukanlah memahami sintaks yang ada, namun untuk memahami bagaimana pola pikir yang tepat untuk melakukan pemrograman.