MODUL 2 PROGRAMMING
Pengenalan C & C++

1. Instalasi Bahasa C

a. Windows
https://www.rose-hulman.edu/class/csse/resources/MinGW/installation.htm

b. Linux
1. Buka terminal
2. Update Package
```
$ sudo apt update
```
3. Install build-essential (sudah termasuk gcc, g++, dan make)
```
$ sudo apt install build-essential
```
4. Cek Instalasi
```
$ gcc --version
```

c. MacOS
1. Buka terminal
2. Install Homebrew
```
$ /bin/bash -c "$(curl -fsSL
  https://raw.githubusercontent.com/Homebrew/install/HEAD/in
  stall.sh)"
```
3. Install gcc using Homebrew
```
$ brew install gcc
```
4. Cek Instalasi
```
$ gcc --version
```

2. Variable and Data Type

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

Pada C, digunakan library stdio.h untuk input dan output. Pertama diperlukan include library:

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

a. Input

```
// format: scanf("<data specifier>", <addr variable>);

/* integer */
scanf("%d", &x);            // 1 variable
scanf("%d %d", &x, &y);     // 2 variable

/* float */
scanf("%f", &z);

/* string */
char str[20];    // Inisialisasi string (array of char)
scanf("%19s", &str);
```

b. Output

```
// format: printf("<format>", <list variable>);

/* contoh */
printf("x = %d\n", x);

// Perhatikan bahwa '\n' adalah new line
```

4. Operators

a. Arithmetic Operators

Operator	Deskripsi	Contoh
+	Penjumlahan 2 operand	`x = 1 + 2;`
–	Pengurangan 2 operand	`x = 1 – 2;`
*	Perkalian 2 operand	`x = 1 * 2;`

/	Pembagian 2 operand	`x = 1 / 2;`
%	Modulus 2 operand	`x = 1 % 2;`
++	Increment	`x++;`
--	Decrement	`x--;`

b. Logical Operators

Relational Operator

Operator	Deskripsi	Contoh
==	Memeriksa apakah kedua operand memiliki value yang sama	`A == B`
!=	Memeriksa apakah kedua operand memiliki value yang beda	`A != B`
>	Memeriksa apakah operand kiri lebih besar dari operand kanan	`A > B`
<	Memeriksa apakah operand kiri lebih kecil dari operand kanan	`A < B`
>=	Memeriksa apakah operand kiri lebih besar atau sama dengan dari operand kanan	`A >= B`
<=	Memeriksa apakah operand kiri lebih kecil atau sama dengan dari operand kanan	`A <= B`

Logical Operator

Operator	Deskripsi	Contoh
&&	and	`A && B`
||	or	`A || B`
!	not	`!A`

c. Assignment Operator

Operator	Deskripsi	Contoh
=	Operator assignment dasar	`x = y + z;`
+=	Operator assignment tambah	`x += 5; // x = x + 5`
-=	Operator assignment kurang	`x -= 5; // x = x - 5`
*=	Operator assignment kali	`x *= 5; // x = x * 5`
/=	Operator assignment bagi	`x /= 5; // x = x / 5`
%=	Operator assignment modulo	`x %= 5; // x = x % 5`

Learn more: https://www.tutorialspoint.com/cprogramming/c_operators.htm

5. If-Else Statement

a. If-Else

Format:

```
if (/* kondisi */) {
      /* aksi */
} else if (/* kondisi */) {
      /* aksi */
} else {
      /* aksi */
}
```

b. Switch case

Format:

```
switch (/* variabel case */) {
      case /* klausa case 1 */:
            /* aksi */
            break
      case /* klausa case 2 */:
            /* aksi */
            break
      case /* klausa case 3 */:
            /* aksi */
            break
      ...
      Default:
            /* aksi default */
            break
}
```

Contoh:

```
int x;      // Deklarasi variabel
scanf("%d", &x);      // Input x
switch (x) {
      case 0: // x == 0
            printf("x adalah bilangan nol");
            // Perhatikan jika tidak menggunakan break
            // Anda dapat mencoba sendiri
      case 1:
            printf("x = 1");
            break
      Default:
            printf("x = %d\n", x);
            break
}
```

6. Loops

a. For loop

Format:

```
while (/* expression */) {
      /* aksi */
}
```

b. While loop

Format:

```
for (/* initial */; */expression*/; /* update */) {
      /* aksi */
}
```

Contoht:

```
for (int i = 0; i < 10; i++) {
      printf("%d\n", i);
}
```

c. Do-While loop

Format:

```
do {
      /* aksi */
} while (/* expression */);
```

7. Function and Procedure

a. Function

Function dapat dimengerti sebagai subprogram yang akan mengembalikan suatu value.

Format:

```
/* return data type */ /* nama function */ (/* param */) {
      /* Aksi function */
      return /* return value */;
}
```

Contoh:

```
int jumlah (int a, int b) {
      return a + b;
}
```

b. Procedure

Procedur dapat dimengerti sebagai subprogram yang akan mengubah suatu state dari program.

Format:

```
void /* nama procedure */ (/* param */) {
      /* Aksi procedure */
}
```

Contoh:

```
void tambah (int *a, int b) {
      *a += b;
}
// Perhatikan bahwa variable masukan a akan berubah valuenya


/* cara pemanggilan */
int a = 0;
tambah(&a, 5);
// a = 5
```

8. Pointer

Pointer berisi alamat mesin. Pointer menunjuk kepada nama yang diacunya, sehingga informasi yang disimpan pada nama dapat diakses. Pointer dapan menunjuk kepada pointer (pointer to pointer). Pointer memungkinkan alokasi dinamik, memori baru di-alokasi berdasarkan kontrol pemrogram, yaitu hanya jika diperlukan. Jika tidak dibutuhkan lagi, ruang memori yang dialokasi dapat di-dealokasi (dikembalikan) ke mesin. Dengan demikian, kontrol pemakaian memori pada saat run time berada di tangan program dan berdasarkan atas kondisi memori saat itu. Di samping keuntungannya, pemrogram harus berhati-hati dan cermat dalam melakukan alokasi/dealokasi. Jika tidak, maka suatu saat memori tidak cukup, atau address mengacu ke suatu yang tidak terdefinisi dan dapat menyebabkan computer “hang”. Pada modul ini hanya akan dijelaskan mengenai pointer sederhana.

Format:

```
<type> *<name>;
```

Contoh:

```
/* Pendefinisian */
int *i;                 /* pointer ke integer */
float *x;               /* pointer ke float */
char *cc;               /* Pointer ke karakter */
FILE *FileKu;           /* Handle sebuah file */
int **argv;             /* pointer ke pointer ke karakter */
int (*tabel)[13];       /* pointer ke array dengan 13 elmn int */
int *tabel[13];         /* pointer ke array dengan 13 elemen yang
                        berptipe pointer ke integer */
/* Alokasi dinamik */
int *iptr;              /* deklarasi dalam kamus */
/* Algoritma */
iptr = (int*) maloc(sizeof(int)); /* alokasi memori */

*iptr=999;              /* nilai yang ditunjuk iptr */
                        /* mengassign array dengan indeks 0 */
```

More reference: https://www.programiz.com/c-programming/c-pointers

9. Contoh Program C sederhana

```
#include <stdio.h>

/* Funtion */
int jumlah (int a, int b) {
      return a + b;
}

/* Procedure */
void tambah (int *a, int b) {
      *a += b;
}

/* Main Program */
Int main () {
      int x, y, z;
      scanf("%d %d", &x, &y);
      z = tambah(x, y);
      printf("%d %d %d\n", x, y, z); // Perhatikan output
      tambah(&x, y);
      printf("%d %d %d\n", x, y, z); // Perhatikan output
      return 0;         // Jangan lupa return
}
```