

Basic Programming Lab Module

This page is intentionally left blank.

Contents

1	Structure of C Programming Language	1
1.1	Goals	1
1.2	Creating new projects on an IDE Code::Blocks	1
1.2.1	Steps to create a new project	1
1.2.2	Exercise	5
1.3	Structure of C Programming Language	5
1.3.1	Exercise	5
1.4	Data Types and Variable	6
1.4.1	Data Types	6
1.4.2	Variable	6
1.4.2.1	Arithmetic and Assignment operators	6
1.4.3	Exercise	7
1.5	Output and Input	8
1.5.1	printf()	8
1.6	scanf	9
1.6.1	Escape Sequence	11
1.6.2	Latihan	12
2	Conditional Instruction	13
2.1	Goals	13
2.2	Logical and Comparison Expression	13
2.2.1	Comparison Expression	13
2.2.2	Logical Expression	13
2.3	If statement	14
2.4	If-else statement	15
2.5	Statement if-else if	16
2.6	Nested if	17
2.7	Exercise	18
3	Loops and Array	19
3.1	Goals	19
3.2	Loop	19
3.2.1	while loop	19
3.2.2	do-while loop	21
3.2.3	for loop	22
3.3	Array	23
3.3.1	Array 1D	23

3.3.2	2D Array and Other Multidimensional Arrays	24
3.4	Example of Using Loops and Array	24
3.4.1	Linear Search	24
3.4.2	Bubble Sort	24
3.5	Exercise	25
4	Fungsi (Subprogram)	27
4.1	Tujuan	27
4.2	Function Declaration	27
4.3	Calling a Function	28
4.4	Function with Arguments	29
4.4.1	Arguments	29
4.4.2	Parameter Passing	30
4.4.2.1	Passing Parameter by Value	30
4.4.2.2	Passing Parameter by Reference	31
4.5	Recursion	32
4.6	Exercise	32

Chapter 1

Structure of C Programming Language

1.1 Goals

- Students are able to create projects on an IDE
- Students can demonstrate his/her knowledge of the structure of a C program
- Students can demonstrate his/her knowledge of C data types
- Students can demonstrate his/her knowledge of C operators
- Students are able to use function to read inputs from keyboard
- Students are able to use function to print texts on screen

1.2 Creating new projects on an IDE Code::Blocks

1.2.1 Steps to create a new project

1. Go to File > New > Project

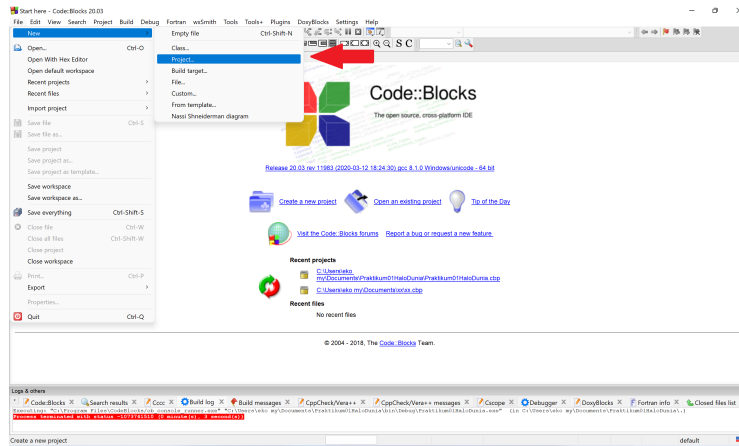


Figure 1.1

2. Click Console Application

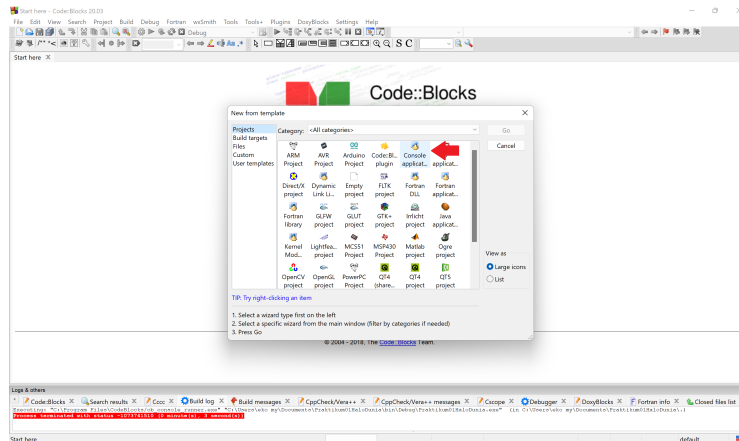


Figure 1.2

3. Choose C as programming language

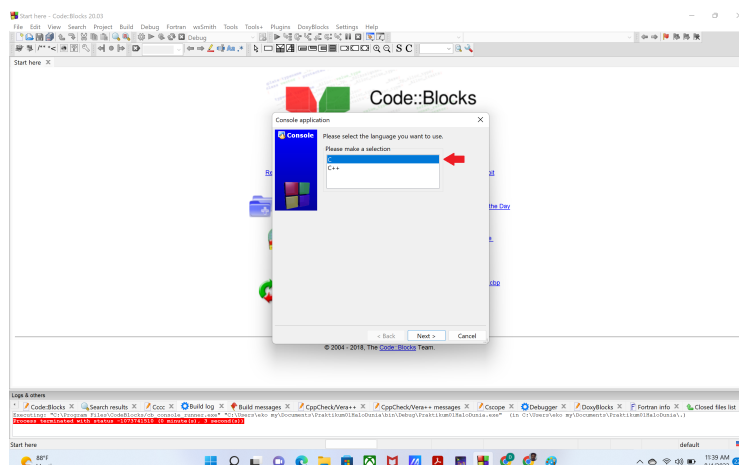


Figure 1.3

4. Give a name to the project

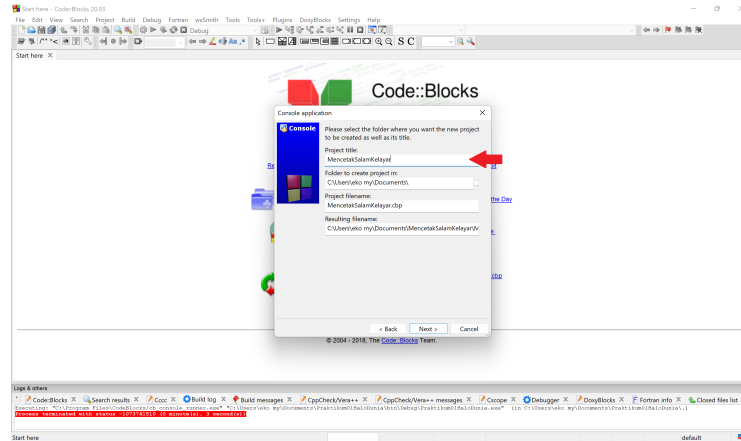


Figure 1.4

5. Choose the compiler (gcc), set the saving directory, and click finish.

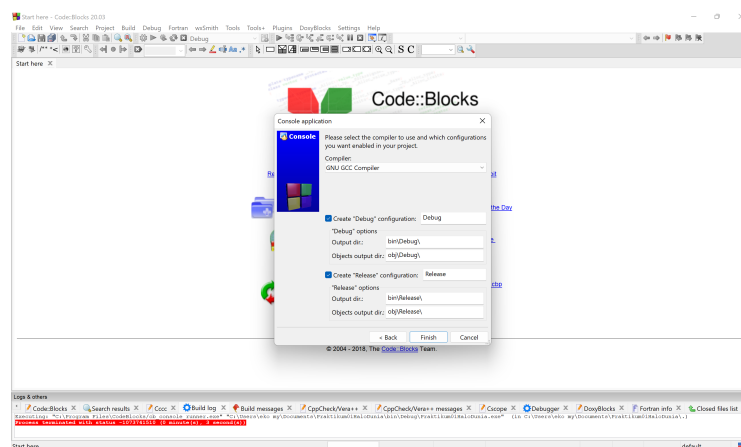


Figure 1.5

6. Type the code in figure 1.6 to the Code::Blocks text editor

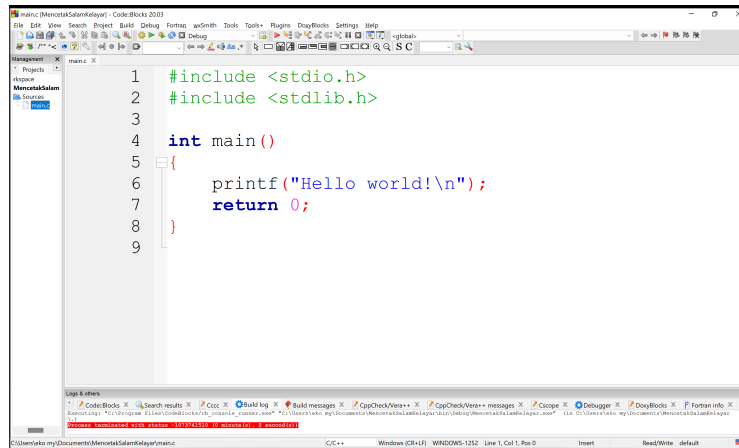


Figure 1.6

7. click Build—>Build and Run or press F9

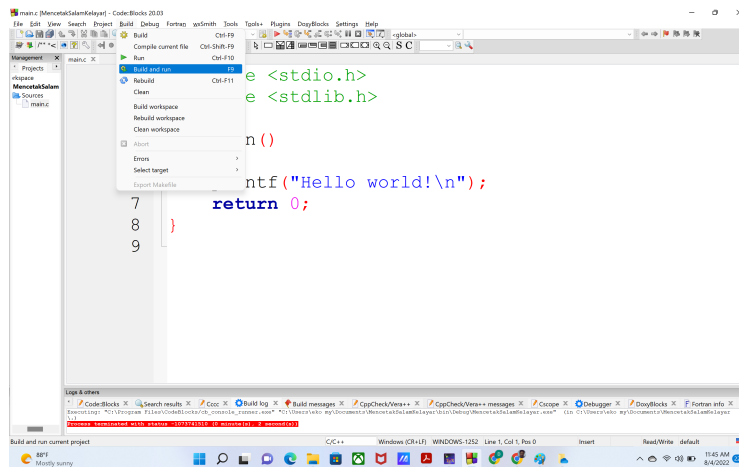


Figure 1.7

8. The program outputs can be seen on the console.

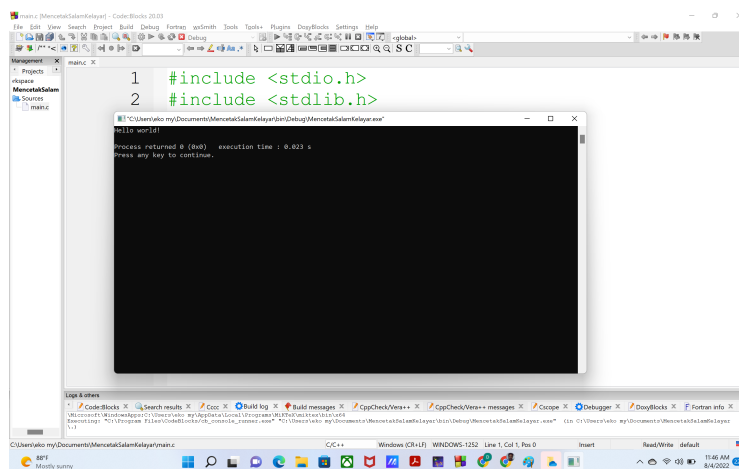


Figure 1.8

1.2.2 Exercise

Create a project with name HaloDunia and write the program like can be seen on figure 1.6 but change Hello World! with Halo Dunia!

1.3 Structure of C Programming Language

Listing 1.1: Contoh program sederhana dalam bahasa C

```
1 #include <stdio.h>
2
3 int main()
4 {
5     //printing to screen
6     printf("Halo Dunia");
7     return 0;
8 }
```

Code on Listing 1.1 is a simple program to print "Halo Dunia" to screen. The following is the explanation what each line of code do in the program.

- Baris 1 : `#include <stdio.h>`
header file library for input and output functions like `printf()` (the one used on line 6)
- Baris 2 : Empty line.
- Baris 3 : `int main()`
The main function. The main function is the first function to be ran when the program starts.
- Baris 4 : `{`
Beginning of the `main()` function code block.
- Baris 5 : `//printing to screen`
Comments. Comments are used to explain what the program is doing. Comments are ignored by the program, but helps the reader.
- Baris 6 : `printf("Halo Dunia");`
Printing "Halo Dunia" to the screen.
- Baris 7 : `return 0;`
Returning the `main()` function (A function ends when it returns)
- Baris 8 : `}`
Closing the `main()` function code block.

1.3.1 Exercise

Try to swap line 6 and line 7 in Listing 1.1. What happened?
What if `return 0;` replaced with `return 1;`?

1.4 Data Types and Variable

1.4.1 Data Types

In C programming language, there are several data types to represent integer, real number, characters, string, and etc.

Table 1.1: Several data types on C

Data Types	Size	Description
int	2 or 4 bytes	saves integers
float	4 bytes	saves real numbers to 8 digit behind decimal point.
double	8 bytes	saves real numbers to 15 digits behind decimal point.
char	1 byte	saves a character

Untuk menampilkan data pada layar, setiap tipe data memiliki format specifier yang dapat digunakan pada formatted string. Berikut adalah format specifier untuk beberapa tipe data. To show the data on screen, every data type has a format specifier that can be used on formatted string. The following is the format specifier for several data types.

Table 1.2: Format Specifier

Format Specifier	Data Type
%d or %i	int
%f	float
%lf	double
%c	char
%s	string

There are still more data types that what was written on Table 1.1. These data types and its specification can be found easily on the internet.

1.4.2 Variable

Variable is a place to save data. Declaring a variable can be done in the following.

Listing 1.2: Deklarasi Variabel C

```
1 DataType VariableName;
```

1.4.2.1 Arithmetic and Assignment operators

Assignment operator can be done to variables that have no `const` or is an l-value variable. Arithmetic operator however can accept both variable with `const` or not (both l-value and r-value). The table below shows some arithmetic operators in C.

Table 1.3: Arithmetic Operators in C

Operator	Nama	Contoh
+	Addition	$x + y$
-	Substraction	$x = y$
*	Multiplication	$x * y$
/	Division	x/y
%	Modulo	$x \% y$

The table below shows some assignment operators.

Table 1.4: Assignment Operators

Operator	Example	Equivalent meaning
=	$x = 5$	$x = 5$
+=	$x += 3$	$x = x + 3$
-=	$x -= 3$	$x = x - 3$
*=	$x *= 3$	$x = x * 3$
/=	$x /= 3$	$x = x / 3$
%=	$x \% = 3$	$x = x \% 3$
&=	$x \& = 3$	$x = x \& 3$
—=	$x \text{ —} = 3$	$x = x \text{ —} 3$
^=	$x \wedge = 3$	$x = x \wedge 3$
>>=	$x >> = 3$	$x = x >> 3$
<<=	$x << = 3$	$x = x << 3$

There is also "shorthands" for some assignments operator like $x+=1$ and $x-1$ which are $++$ and $--$ that is called increment and decrement respectively. These shorthands are used like the following.

```
x++;
x--;
++x;
--x;
```

1.4.3 Exercise

Listing 1.3: Using assignment operator on const variable

```
1 #include <stdio.h>
2 int main()
3 {
4     //deklarasi variabel const
5     const int x=0;
6     x=1;
7     return 0;
8 }
```

Try to compile the program in Listing 1.3, what happened?

1.5 Output and Input

1.5.1 printf()

`printf` is a function in C that is used to print formatted string. You can use format specifier within the formatted string to outputs your variables.

```
printf(const char *format,v1,v2,...,vn)
```

The format specifier for each data types can be seen on Table [1.2](#)

Contoh 1.5.1 Printing text to the screen.

Listing 1.4: Mencetak Tulisan "Pemrograman C Ke layar

```

1  #include <stdio.h>
2  int main()
3  {
4      // Printting the text written within the " symbol
5      printf("C Programming");
6      return 0;
7  }
8
```

- All C program must have `main()` function where the program needs to run the code.
- `printf()` function is a function from `stdio.h` library. This function outputs the string inside the symbol " to the screen.
- `return 0;` statement in the `main()` function tells the program to exit.

Contoh 1.5.2 Printing integer.

```

1  #include <stdio.h>
2  int main()
3  {
4      int testInteger = 5;
5      printf("Number = %d", testInteger); // <- %d format
6      return 0;
7  }
8
```

The code above uses the format specifier `%d` to prints `int` data type. The `%d` part of the string will be replaced with the value of `testInteger`.

Contoh 1.5.3 Output real numbers (float or double)

- Base : using float data type.
- Height: using float data type.
- Area : using float data type.

- Triangle area formula:

$$Luas = \frac{1}{2} \times Alas \times Tinnggi \quad (1.1)$$

```

1  #include <stdio.h>
2
3  int main()
4  {
5      // declare the variables
6      float Base;
7      float Height;
8      float Area;
9      // Initialize the values
10     Base = 10;
11     Height = 5;
12     // Calculate the area
13     Area = 0.5*Base*Height;
14     //Print the area to the screen
15     printf("Area = %f",Area);
16     return 0;
17 }
18
19

```

Explanation:

Line 6-8 Declaring the variables needed to save the data about the triangle.

Line 10 dan 11 Giving values to the variables

Line 13 Calculate the area according to the formula

Line 15 Prints the area with printf and %f format specifier

.

1.6 scanf

`scanf(const char*format, ...)` reads input according to the format string.

1. Syntax

```
scanf(const char *format, ...)
```

2. Parameter

Format string in C consist of one or more whitespace, non-whitespace, and format specifiers.

3. Return Value

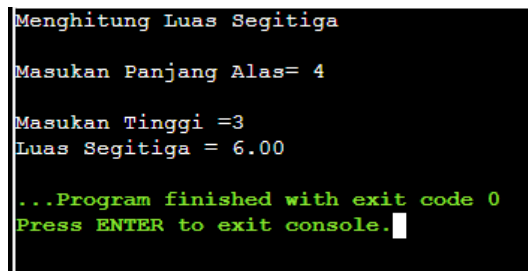
The function will return the number of arguments it has sucessfully read.

1.6.1 Calculating the area of an triangle with the base Alas dan height Tinggi is inputted from keyboard

```

1  #include <stdio.h>
2
3  int main()
4  {
5      float Alas ,Tinggi,Luas;
6
7      printf("Calculating the area of an triangle\n");
8      printf("\nEnter the base length= ");
9      scanf("%f",&Alas);
10     printf("\nEnter the height=");
11     scanf("%f",&Tinggi);
12     Luas = 0.5*Alas *Tinggi;
13     printf("Triangle area = %.2f", Luas);
14     return 0;
15 }
16

```



```

Menghitung Luas Segitiga
Masukan Panjang Alas= 4
Masukan Tinggi =3
Luas Segitiga = 6.00
...Program finished with exit code 0
Press ENTER to exit console.

```

Figure 1.9

Line 9 `scanf("%f",&Alas);` ask for the input for the the triangle base

Line 11 `scanf("%f",&Tinggi);` ask for the input for the triangle height

Line 13 `printf("Triangle Area = %.2f", Luas);`, The `.2` in `%.2f` tells the program to outputs only 2 digits after the decimal point.

Contoh 1.6.2 Program to input name and email.

This example shows how to input string or text from keyboard and outputs it on the screen. Input from this program consist of `sName` and `sEmail`.

```

1  #include <stdio.h>
2
3  int main ()
4  {
5      char sName[20], sEmail[30];
6
7      printf("Insert Name: ");
8      scanf("%19s", sName);
9
10     printf("Insert Email : ");
11     scanf("%29s", sEmail);
12
13     printf("Name : %s\n", sName);
14     printf("Email:%s", sEmail);
15     return(0);
16 }
17

```

1.6.1 Escape Sequence

Some characters can't be written on the format string because they are used to format the outputs. So, to outputs those special characters we use escape sequences.

Table 1.5: Escape Sequence

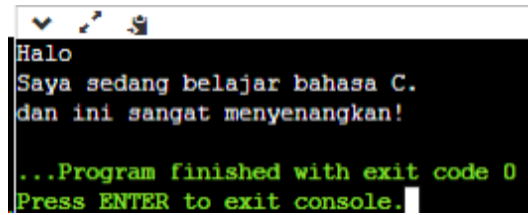
Escape sequence	What it will output	
\a	bell, alarm	
\b	Backspace	
\f	Ganti halaman	
\n	Ganti baris	
\r	Carriage return	
\t	tab horisontal	
\v	tab vertikal	
\'	Petik tunggal	
\"	Petik Ganda	
\?	Tanda tanya	
\\	Backslash	

Example 1.6.1 Changing line with the escape sequence \n.

```

1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Halo \nSaya sedang belajar bahasa C.\ndan ini
6          sangat menyenangkan!");
7      return 0;
8  }

```



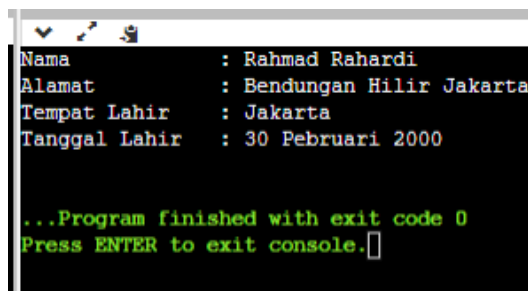
```
Halo
Saya sedang belajar bahasa C.
dan ini sangat menyenangkan!

...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 1.10

Contoh 1.6.1 Using escape sequence `\t` setting the tab.

```
1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Nama \t\t: Rahmad Rahardi\n");
5     printf("Alamat \t\t: Bendungan Hilir Jakarta\n");
6     printf("Tempat Lahir \t: Jakarta\n");
7     printf("Tanggal Lahir \t: 30 Pebruari 2000\n");
8
9     return (0);
10 }
```



```
Nama           : Rahmad Rahardi
Alamat          : Bendungan Hilir Jakarta
Tempat Lahir    : Jakarta
Tanggal Lahir   : 30 Pebruari 2000

...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 1.11

1.6.2 Latihan

Cobalah buat suatu program yang dapat menerima input berupa nama dan NRP kemudian menampilkannya pada layar.

Chapter 2

Conditional Instruction

2.1 Goals

- Students know and able to utilize logical and comparison expression in C programming language.
- Students know and able to utilize conditional instruction syntaxes in C programming language.

2.2 Logical and Comparison Expression

2.2.1 Comparison Expression

Berikut adalah operator-operator yang digunakan pada suatu ekspresi perbandingan The following are the operators used in comparison expressions.

Table 2.1: Comparison Operators

Operator	Name	Example Expression
==	Equals	x == y
!=	Not Equals	x != y
>	Greater than	x > y
<	Lesser than	x < y
>=	Greater or Equal than	x >= y
<=	Lesser or Equal than	x <= y

A Comparison Expression will return boolean value **true** or **false** which is also represented with the value 1 or 0. As example:

```
printf("%d",0>1); // will output 0 to the screen
printf("%d",0<1); // will output 1 to the screen
```

2.2.2 Logical Expression

Berikut adalah operator-operator logika yang digunakan pada suatu ekspresi logika The following are the logical operators used on a Logical Expression

Table 2.2: Logical Expression

Operator	Name	Example Expression
&&	AND	$x < 5 \ \&\& \ x < 10$
	OR	$x < 5 \ \ x < 4$
!	NOT	$!(x < 5 \ \&\& \ x < 10)$

Like comparison expression, logical expression will return boolean values.

2.3 If statement

if statement is used to decide which block of code to be executed if the condition is true.

```
// Block of code before if
if (Condition)
{
    // Block of code to be executed if the condition is true.
}
// block of code after if
```

Sebagai contoh, perhatikan program berikut As example, look at the following code

Listing 2.1: If statement example

```
1  include <stdio.h>
2
3  int main()
4  {
5      //Deklarasi variabel
6      int uangSaya, hargaRoti;
7      uangSaya = 5000;
8      hargaRoti = 10000;
9
10     if (uangSaya >= hargaRoti)
11     {
12         printf("saya bisa beli roti\n");
13     }
14     printf("hehe");
15     return 0;
16 }
```

This program outputs

```
hehe
```

If line 7 changed to `uangSaya=10000`, the outputs of the program would be

```
saya bisa beli roti
hehe
```

2.4 If-else statement

Statement else digunakan untuk menentukan blok kode yang di jalankan apabila kondisi salah. Else statement is used to decide the block of code to be executed if the condition is false.

```
// Block of code before if
if (Condition)
{
// Block of code to be executed if the condition is true
} else
{
// Block of code to be executed if the condition is false
}
// Block of code after if-else statement
```

The following is an example of using if-else statement:

Listing 2.2: if-else example

```
1  include <stdio.h>
2
3  int main()
4  {
5      //Deklarasi variabel
6      int uangSaya, hargaRoti;
7      uangSaya = 5000;
8      hargaRoti = 10000;
9
10     if (uangSaya >= hargaRoti)
11     {
12         printf("saya bisa beli roti\n");
13     }
14     else
15     {
16         printf("saya tidak bisa beli roti\n");
17     }
18     printf("hehe");
19     return 0;
20 }
```

The output of the program is

```
saya tidak bisa beli roti
hehe
```

If line 7 changed to `uangSaya=10000`, the outputs of the program would be

```
saya bisa beli roti
hehe
```

2.5 Statement if-else if

Statement `else if` digunakan untuk menjalankan blok kode apabila kondisi statement `if` atau `else if` sebelumnya bernilai salah. The `else if` statement is used to run a block of code when the condition in `if` or the previous `else if` is false.

```
// block of code before if
    if (Condition1)
    {
        /* block of code to be executed if Condition1
        is true*/
    }
    else if (Condition2)
    {
        /* block of code to be executed if Condition1 is false
        and Condition2 is true */
    }
    else if (Condition3)
    {
        /* Block of code to be executed when
        Condition1 and Condition2 is false and
        Condition3 is true*/
    }
    ...
    else if (ConditionN)
    {
        /* Block of code to be executed when
        Condition1 to ConditionN-1 is false and
        ConditionN is true*/
    }
    else
    {
        /* Block of code to be executed when
        Condition1 to ConditionN is false*/
    }
// Block of code after if
```

Berikut contoh penggunaan if-else if

Listing 2.3: if-else if example

```
1  include <stdio.h>
2
3  int main()
4  {
5      //Deklarasi variabel
6      int uangSaya, hargaRoti;
7      uangSaya = 5000;
8      hargaRoti = 10000;
```

```
9
10     if (uangSaya>hargaRoti)
11     {
12         printf("saya bisa beli roti\n");
13     }
14     else if(uangSaya==hargaRoti)
15     {
16         printf("saya bisa beli roti tapi uang saya akan
17 langsung habis\n");
18     }
19     else
20     {
21         printf("saya tidak bisa beli roti\n");
22     }
23     printf("hehe");
24     return 0;
25 }
```

The output of this program is

```
saya tidak bisa beli roti
hehe
```

If line 7 changed to `uangSaya=10000`, the output of the program would be

```
saya bisa beli roti tapi uang saya akan langsung habis
hehe
```

If line 7 changed to `uangSaya=12000`, the output of the program would be

```
saya bisa beli roti
hehe
```

2.6 Nested if

Nested if is when there is a conditional statements within a block of code inside the conditional statement

```
// Block of code before if
if (Condition1)
{
    if (Condition2)
    {
        // do something
    }
    else
    {
        // do another thing
    }
}
else
```

```
{  
    // do something else  
}
```

Below is an example of using nested if

Listing 2.4: nested if example

```
1  include <stdio.h>  
2  
3  int main()  
4  {  
5      // Declare the variables  
6      int myMoney,breadPrice, friendsMoney;  
7      myMoney = 5000;  
8      breadPrice = 10000;  
9      friendsMoney = 42069;  
10  
11  
12     if (myMoney>breadPrice)  
13     {  
14         printf("I can buy bread\n");  
15     }  
16     else if(myMoney==breadPrice)  
17     {  
18         printf("I can buy bread but I will ran out of money\n")  
19     }  
20     else  
21     {  
22         if(friendsMoney+myMoney >= breadPrice)  
23         {  
24             printf("I can buy bread if I borrow my friend money  
25             \n");  
26         }  
27         else  
28         {  
29             printf("I can't buy bread\n");  
30         }  
31     }  
32     printf("hehe");  
33     return 0;  
34 }
```

2.7 Exercise

Try to make a program that receives 3 integer input A, B, and C. Then outputs those 3 integers to the screen sorted from smallest to largest. Do this only using conditional statements.

Chapter 3

Loops and Array

3.1 Goals

- Students are able to use while-loop on C
- Students are able to use do-while loop on C
- Students are able to use for loop on C
- Students are able to use one dimensional or multidimensional array
- Students are able to use loops to process data on arrays

3.2 Loop

3.2.1 while loop

While loop will run the code block within it repeatedly as long as the loop condition is true

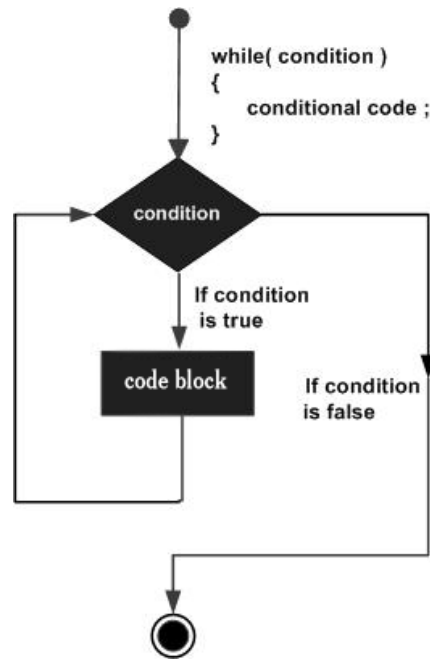


Figure 3.1: Flow chart of while loop

Its syntax in C programming language is as follows

```

while(Condition)
{
    // The block of code that will be repeated
}

```

As an example, look at the following code

Listing 3.1: Contoh Penggunaan while

```

1 int main()
2 {
3     int uangSaya, hargaRoti;
4     uangSaya = 10000;
5     hargaRoti = 2000;
6     while(uangSaya >= hargaRoti)
7     {
8         printf("Beli roti 1, uang saya sisa %d", uangSaya -
9             hargaRoti);
10        uangSaya -= hargaRoti;
11    }
12    printf("Uang saya tidak cukup lagi");
13    return 0;
14 }

```

The output of the program in Listing 3.1 are

```

Beli roti 1, uang saya sisa 8000
Beli roti 1, uang saya sisa 6000
Beli roti 1, uang saya sisa 4000
Beli roti 1, uang saya sisa 2000

```



```
Beli roti 1, uang saya sisa 0
Uang saya tidak cukup lagi
```

You can see the line 9 of the code causes the variable `uangSaya` to have its value subtracted by 2000 for every loop until `uangSaya` is no longer greater than equal to `hargaRoti`. The loop condition will be invalid and finally exits the loop. Then it prints "Uang saya tidak cukup lagi", the command after the while loop statement.

3.2.2 do-while loop

do-while loop is very similar to while loop. The only difference is that do-while loop will execute the code block inside it once, and then checks the condition.

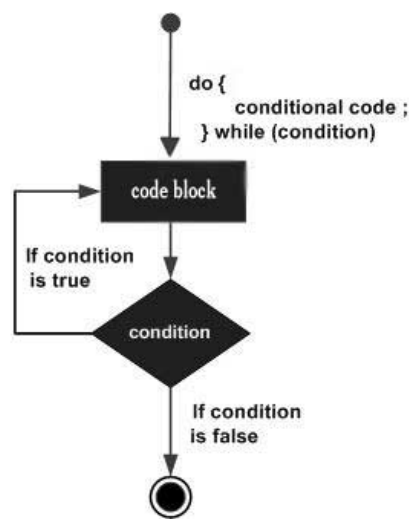


Figure 3.2: do-while flow chart

Its syntax in C is as follows:

```
do{
    // the block of code that will be repeated
}while(Condition)
```

Look at the following example.

Listing 3.2: Contoh Penggunaan do-while

```

1 int main()
2 {
3     int uangSaya, hargaRoti;
4     uangSaya = 10000;
5     hargaRoti = 12000;
6     do{
7         printf("Beli roti 1, uang saya sisa %d", uangSaya -
8             hargaRoti);
9         uangSaya -= hargaRoti;
10    }while(uangSaya >= hargaRoti)
11    printf("Uang saya tidak cukup lagi");

```

```

11  return 0;
12  }

```

The output of the code above are

```

    Beli roti 1, uang saya sisa -2000
    Uang saya tidak cukup lagi

```

The variable `uangSaya` is subtracted by `hargaRoti` before checking the `uangSaya >= hargaRoti` condition. Had the code above uses while loop, the repeating block of code wouldn't have executed even once.

3.2.3 for loop

If you have a block of code like this:

```

InitializationStatement; // e.g.: int i = 0;
while(Condition){
    // do something
    updateStatement; // e.g.: i++
}

```

This is equivalent to

```

for(InitializationStatement;Condition;updateStatement){
    // do something
}

```

As example, look at the following code:

Listing 3.3: Contoh Penggunaan for

```

1  int main()
2  {
3      int i=0;
4      for(i=1;i<10;i++){
5          printf("%d ",i);
6      }
7      return 0;
8  }

```

The output of this program are

```

1 2 3 4 5 6 7 8 9

```

The following is the code if code in Listing 3.3 converted to its while-loop form

Listing 3.4: For dalam bentuk while

```

1  int main()
2  {
3      int i=0;
4      i=1;
5      while(i<10){
6          printf("%d ",i);

```

```
7         i++;  
8     }  
9     return 0;  
10 }
```

3.3 Array

Array is a collection of data where each element of it has the same name(indexed) and data type. Every element in an array can be accessed using its element index.

3.3.1 Array 1D

One dimensional array variable can be declared by deciding the data type of the element and the number of element that is needed.

Syntax:

```
DataType variableName [arraySize];
```

1. **DataType.**

The data type of the elements in the array, e.g. `float`, `int`, etc.

2. **variableName**

variableName follows the variable naming convention

3. **arraySize**

Integer more than 0. Defining the number of element an array has.

Initializing one dimensional array can be done like shown below:

```
int contoh_array[5] = {4,2,0,6,9};
```

Data in an array can be accessed by using an integer that is the index of the array. Look at the code below

Listing 3.5: Contoh Mengakses Array 1D

```
1 int main()  
2 {  
3     int arr[5] = {4,2,0,6,9};  
4     printf("%d\n",arr[0]);  
5     printf("%d\n",arr[4]);  
6     int i = 0;  
7     printf("%d\n",arr[i]);  
8     for(i=0;i<5;i++)  
9         printf("%d",arr[i]);  
10 }
```

The code in Listing 3.5 will give output

```
4  
9  
4  
42069
```

3.3.2 2D Array and Other Multidimensional Arrays

2D array is basically a 1D array of 1D array. Intuitively, you can define a 2D array like as seen below:

```
DataType variableName[arraySize1][arraySize2];
```

This also applies to multidimensional array.

```
DataType variableName[arraySize1]...[arraySizeN];
```

There will be $arraySize_1 \times arraySize_2 \times \dots \times arraySize_n$ of elements that would be allocated to the memory after doing multidimensional array like that.

To initialize multidimensional array, you can do the following:

```
int arr[2][2] = {{1,2},{3,4}};
```

3.4 Example of Using Loops and Array

3.4.1 Linear Search

Linear Search is a technique to search for an element by visiting each element one by one. The following code is an example of how to find the largest number in an array using linear search.

Listing 3.6: Finding the largest integer

```
1 int main()
2 {
3     int arr[5] = {4,2,0,6,9};
4     int currentMax = arr[0];
5     int i;
6     for(i=0;i<5;i++){
7         if(currentMax < arr[i]){
8             currentMax = arr[i];
9         }
10    }
11    printf("Bilangan terbesar adalah %d",currentMax);
12 }
```

3.4.2 Bubble Sort

Bubble Sort is a simple algorithm to sort data. The bubble sort algorithm can be seen below:

Listing 3.7: Bubble Sort

```
1 int main()
2 {
3     int arr[5] = {4,2,0,6,9};
4     int tmp;
5     for(int i=0;i<5;i++){
6         for(int j = 0; j < 5-i-1;j++){
```

```
7         if(arr[j]>arr[j+1]){
8             tmp = arr[j];
9             arr[j] = arr[j+1];
10            arr[j+1] = tmp;
11        }
12    }
13 }
14
15 for(int i=0;i<5;i++){
16     printf("%d ",arr[i]);
17 }
18 }
```

3.5 Exercise

- Try to initialize a multidimensional array with for loop
- create a program to fill the data of an array by keyboard input.
- What would happen if an array `arr` is accessed with `arr[-1]`?
- What would happen if an array `arr` with size 5 is accessed with `arr[5]`?
- Look at the following code

```
for(i=0;i<10;i++){
    for(j=i;j<10;j++){
        printf("A");
    }
}
```

How many "A" will be printed on the screen if that block of code is executed?

This page is intentionally left blank.

Chapter 4

Fungsi (Subprogram)

4.1 Tujuan

- Students understand how to create and call functions in C
- Students able to pass parameter by value and by reference in C.
- Students understand and able to apply recursion in C.

The advantages of using functions in C programming language are:

- Some code snippets can be reusable when using functions.
- We can call C functions any number of times in a program and from any place in a program.
- Large C codes can be splitted to several function, thus easier to track.

4.2 Function Declaration

Every C program has atleast one function, that is the `main()` function. You can also define functions other than `main()`.

Syntax :

```
return_type function_name( parameters list){  
    // function body  
    return something;  
}
```

- Return Type.
The data type a function has to return.
- `function_name`.
The name of the function
- parameters list.
The parameters of the function.

- **Badan fungsi.**
The block of code that will be executed when the function is called.
- **return something;**
A statement to return a value (**something**). Returning the function causes the function to end. For functions that doesn't return a value (void type function), ending the function can be done by using **return;**

Example

```

1 float TriangleArea(float Base, float Height)
2 {
3     float Area;
4     Area = 0.5*Base*Height;
5     return Area;
6 }

```

4.3 Calling a Function

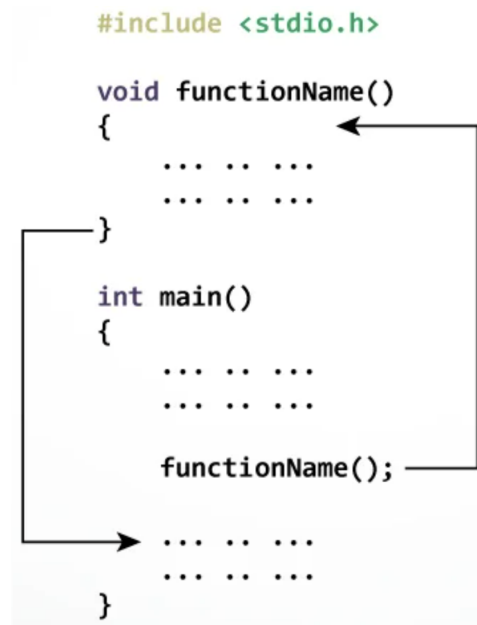


Figure 4.1

```

1  #include <stdio.h>
2  // Mendeklarasikan fungsi luasSegitiga
3  // Parameter input ALas , dan Tintgi
4  // Output float
5  float TriangleArea(float Base, float Height)
6  {
7      float Area;
8      Area = 0.5*Base*Height;
9      return Area;
10 }

```



```
11 int main()  
12 {  
13     float Bs = 4,Hg=10,L;  
14     // Calling the TriangleArea function  
15     L=TriangleArea(Bs,Hg);  
16     printf("Area = %f",L);  
17     return 0;  
18 }
```

1. Line 5-10: Defining the function `Triangle Area` with

- Two input parameter :
input Base and Height with float data type.
- Single valued output with float data type.

4.4 Function with Arguments

4.4.1 Arguments

1. **Parameter :**

- (a) Parameters are the variable in the function that points to the part of the data that is inputted to the function.
- (b) These data is called arguments.

2. **Formal Parameter:**

- (a) Parameter that is written within the function definition is called formal parameter.
- (b) Formal Parameter is always a variable, Actual Parameter however doesn't necessarily has to be a variable.

3. **Actual Parameter:**

- (a) Parameter that is used when calling the function
- (b) Actual Parameter could take the form of number, expression, or another function call.

```

#include <stdio.h>
return_type func_name(arguments);
{
    .....
    .....
}

Int main()
{
    .....
    func_name(arguments_value);
    .....
    return 0;
}

```

Figure 4.2

4.4.2 Parameter Passing

To use a function with parameter, the parameters must be passed to the function first. In general, there are two ways to pass parameter to a function

- Pass parameter by value, pass the value of the variable to the function.
- Pass parameter by reference, pass the reference of a variable (its memory address) to the function.

4.4.2.1 Passing Parameter by Value

Listing 4.1: Passing by Value

```

1 #include <stdio.h>
2 int swapAndReturnTheSum(int x, int y) {
3     int z;
4     z = x;
5     x = y;
6     y = z;
7     return x+y;
8 }
9 int main()
10 {
11     int a = 1;
12     int b = 2;
13     int sum = swapAndReturnTheSum(a,b);

```

```
14     printf("sum: %d\n",sum);
15     printf("values of a dan b now:\n");
16     printf("a: %d\n",a);
17     printf("b: %d\n",b);
18 }
```

line 3-6 of the code in Listing 4.1 is a set of assignments to swap the values of 2 variable. However, when the program is executed, the output would be the following.

```
sum: 3
values a and b now:
a: 1
b: 2
```

The values of a and b did not swap. When passing parameter by value, anything that is done within the function body will have no effect on the parameter that is "passed on" the function. The value of the actual parameter will be assigned to the formal parameter, so we are not doing operation directly on the actual parameter.

4.4.2.2 Passing Parameter by Reference

Look at line 2 of the following code.

Listing 4.2: Passing by Reference

```
1  #include <stdio>
2  int swapAndReturnTheSum(int &x, int &y) {
3      int z;
4      z = x;
5      x = y;
6      y = z;
7      return x+y;
8  }
9  int main()
10 {
11     int a = 1;
12     int b = 2;
13     int sum = swapAndReturnTheSum(a,b);
14     printf("sum: %d\n",sum);
15     printf("values of a dan b now:\n");
16     printf("a: %d\n",a);
17     printf("b: %d\n",b);
18 }
```

When this program is executed, it will output the following.

```
sum: 3
values of a and b now
a: 2
b: 1
```

When the function `swapAndReturnTheSum(a,b)` is called, the memory address of `a` and `b` is passed on to the function. Therefore, in line 4-6, the `x` and `y` will point to the memory of the actual parameter that is inputted in line 13, so we are doing assignments directly to the actual parameter. When passing by reference, we can't call the function with parameter that has no memory address. As example `swapAndReturnTheSum(1,2)` cannot be done as the number 1 and 2 doesn't have memory address.

4.5 Recursion

Recursion is when a function calls itself within its function body. As an example, look at the code below.

Listing 4.3: Factorial with recursion

```

1 int factorial(int n) {
2     if (n==1)
3         return 1;
4     return n*factorial(n-1);
5 }
```

The factorial function calls another factorial function in line 4. Initially, the function $factorial(n)$ is called. This function however will return $n \times factorial(n-1)$, then $factorial(n-1)$ akan mengembalikan $(n-1) \times factorial(n-1-1)$. Eventually it became like this:

$$\begin{aligned}
 factorial(n) &= n \times factorial(n-1) \\
 &= n \times (n-1) \times factorial(n-2) \\
 &= n \times (n-1) \times (n-2) \times \cdots \times 2 \times factorial(1) \\
 &= n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1
 \end{aligned}$$

4.6 Exercise

- Create a function that can take 2 integer `a` and `b` then returns a^b
- Create program that do the bubble sort algorithm but the process of swapping 2 elements in the array is done with a function.
- What problems that can be solved easier with functions?