

Basic Programming Practicum

C Programming Introduction

2023

1 General Purposes

- Students can create projects within the IDE.
- Students can demonstrate their knowledge about program structure in the C language
- Students can demonstrate their knowledge about data types in the C language
- Students can demonstrate their knowledge of data types in the C language
- Students are able to use functions to read input from the keyboard
- Students are able to use functions to print text on the screen

2 Introduction to C Programming Language

The C language was developed by Dennis M. Ritchie and Brian W. Kernighan in the early 1970s. There are several standards for the C programming language. There are several guidelines for writing C programming language. Below are some of the standards:

1. Kernighan & Ritchie Definition(K&R)
2. ANSI-C (X-3.159 -1989-)
3. AT&T (for superset C, C++) definition, and
4. GNU Coding Standards

Implementation and uses of the C programming language

1. Creating operating systems and it's system programs
2. Programmin language that is "very close" to hardware (e.g., for device control).
3. Developing toolkits
4. Writing application programs

3 IDE (Integrated Development Environment)

IDE stands for "Integrated Development Environment" in English. In Bahasa Indonesia, IDE can be translated as "Lingkungan Pengembangan Terintegrasi" or "Ruang Kerja Pengembangan Terpadu." IDE is a software designed to assist software developers in the process of development, coding, and testing computer applications.

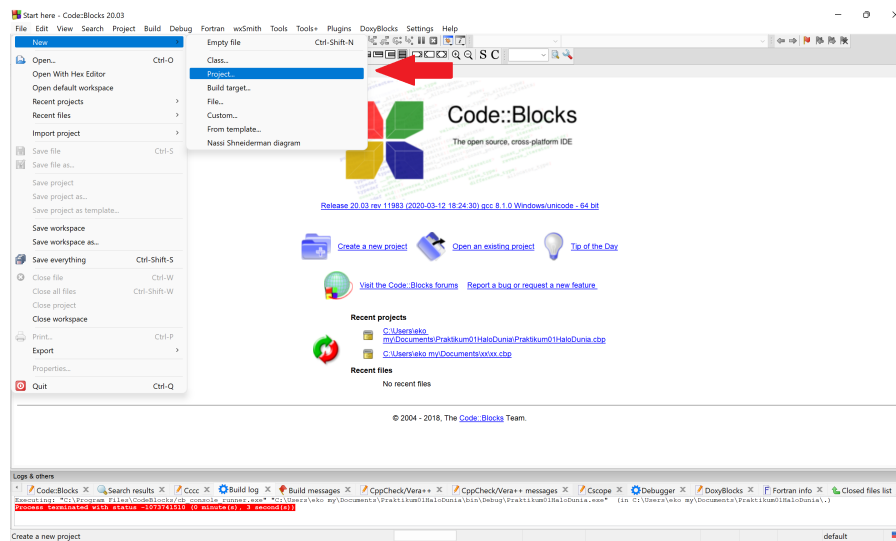
Here are several list of C programming language IDE applications that can be used.

- Code::Blocks
- DevC++

4 Creating new project in IDE Code::Blocks

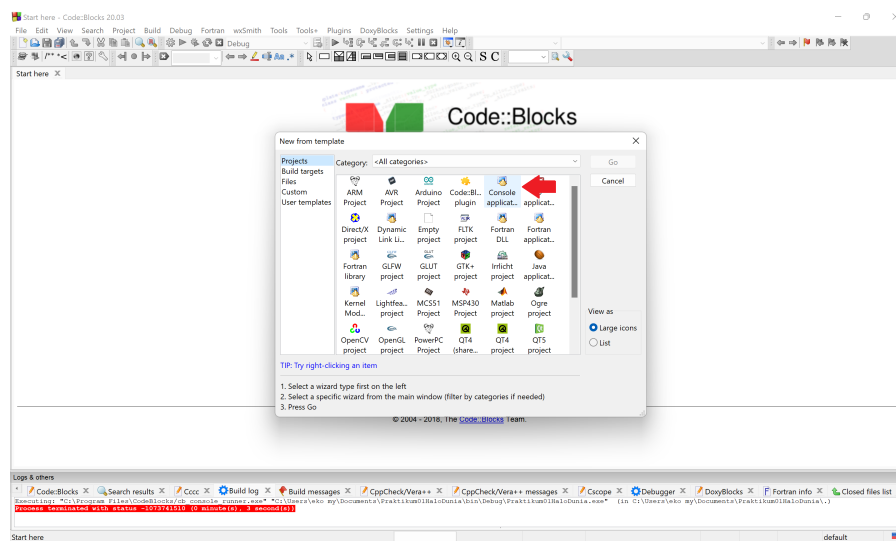
4.1 Steps to create a new project

1. Go to File > New > Project



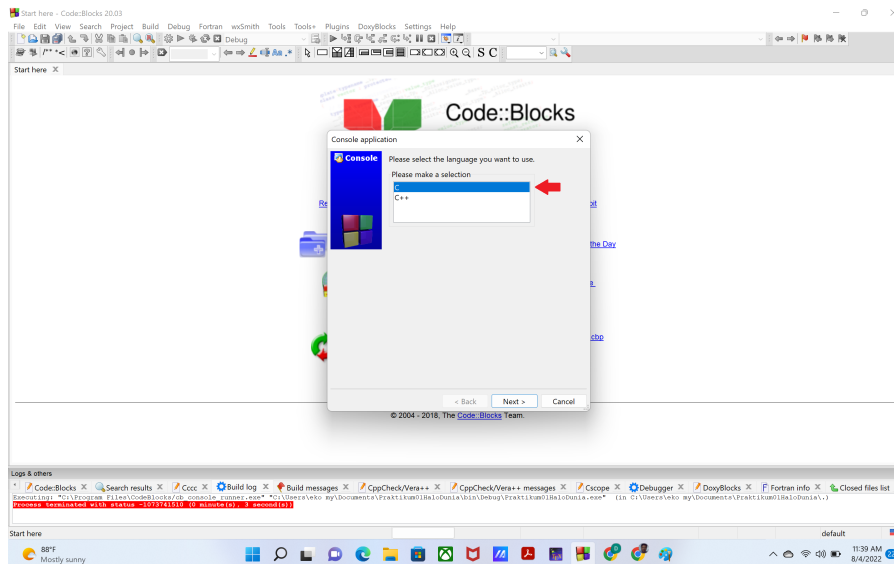
Gambar 1

2. Click on Console Application



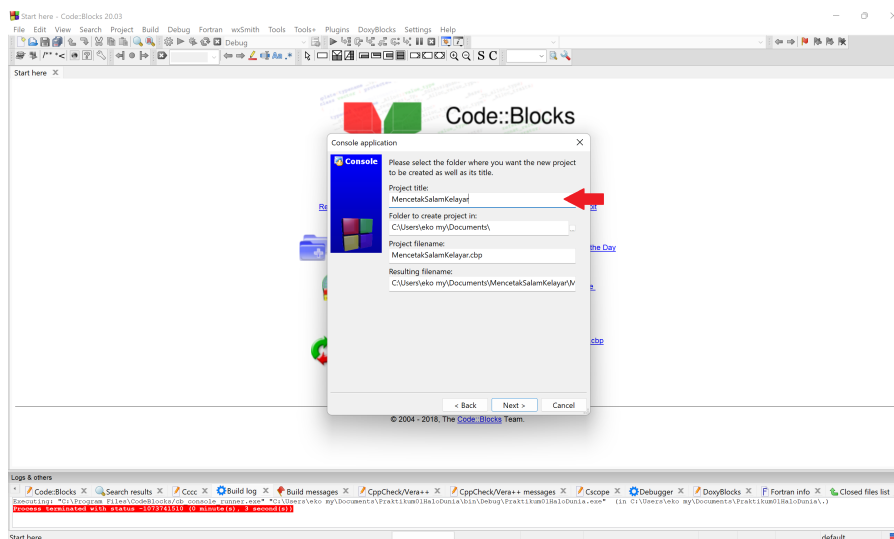
Gambar 2

3. Choose C as the programming language



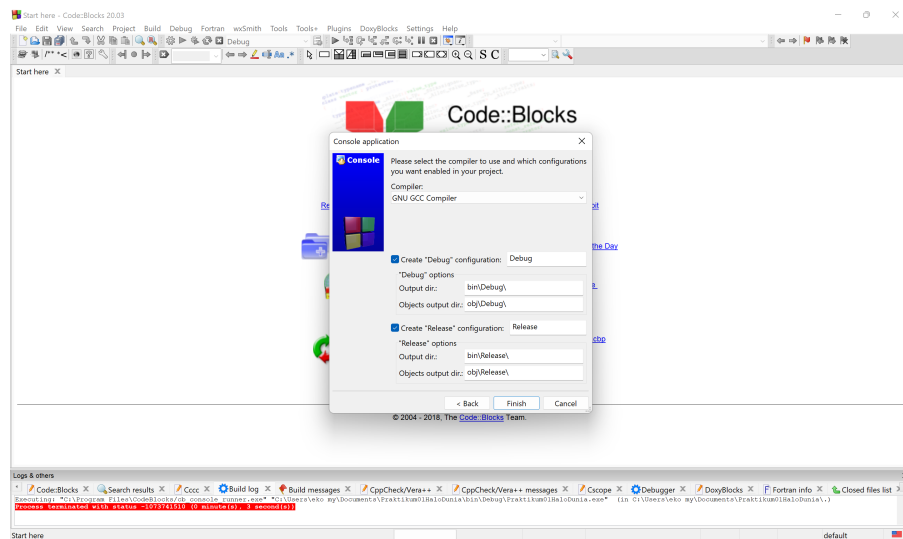
Gambar 3

4. Insert your project name



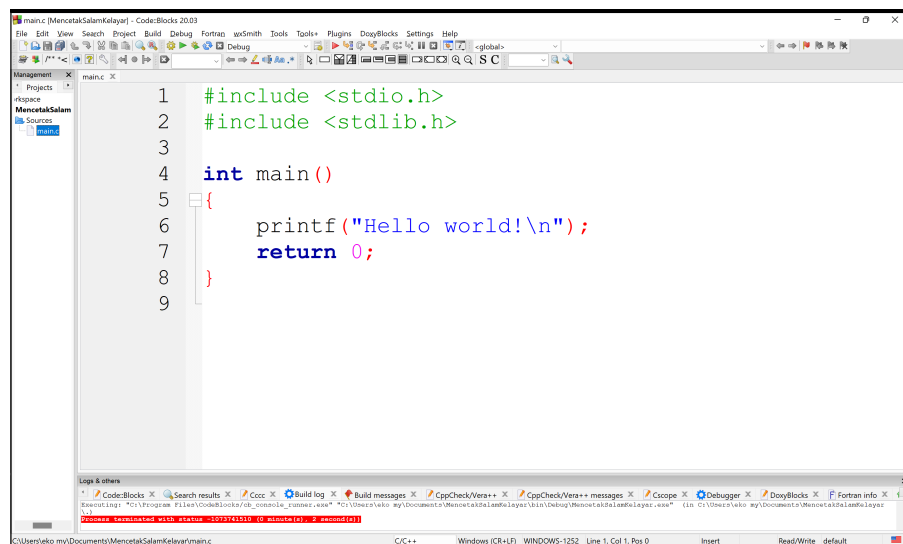
Gambar 4

5. Choose the compiler (gcc), select a directory to save your project, and click save



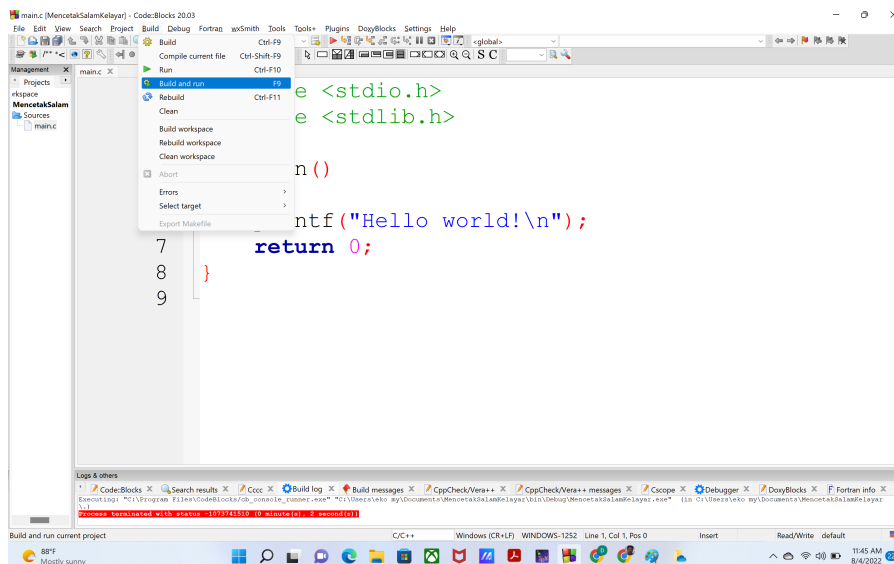
Gambar 5

6. Write your code 6 in Code::Blocks



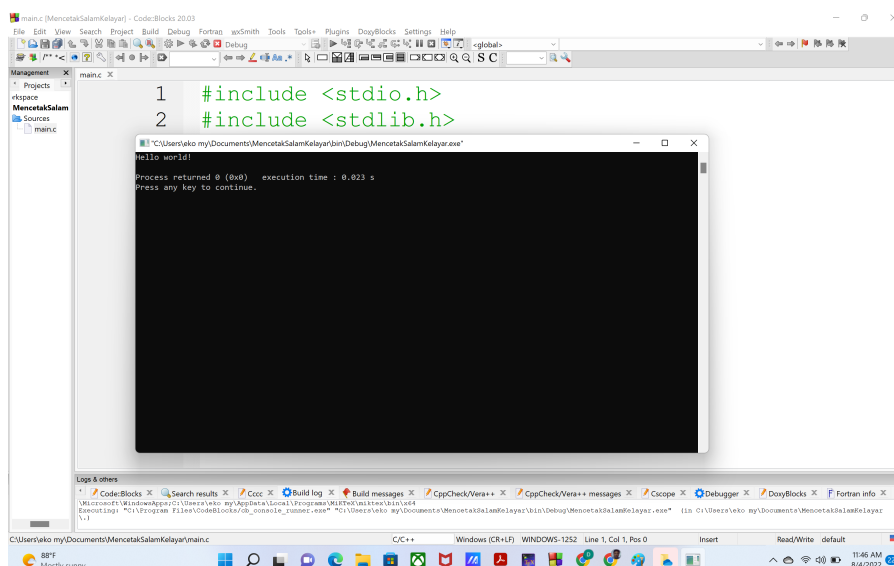
Gambar 6

7. Click Build—>Build and Run or press F9 on your keyboard



Gambar 7

8. The program output can be seen on the console tab



Gambar 8

4.2 Exercise

1. Create a project with name HaloDunia and write the program like can be seen on figure 6 but change Hello World! with Halo Dunia!

5 Structure of C Programming Language

Listing 1: Simple program example for C programming language

```
1 #include <stdio.h>
2
3 int main()
4 {
```

```

5  //printing to screen
6  printf("Halo World");
7  return 0;
8  }

```

Code on Listing 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.

Row 1 : `#include <stdio.h>`

header file library for input and output functions like `printf()` (the one used on line 6)

Row 2 : Empty line.

Row 3 : `int main()`

The main function. The main function is the first function to be ran when the program starts.

Row 4 : `{`

Beginning of the `main()` function code block.

Row 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.

Row 6 : `printf("Halo Dunia");`

Printing "Halo Dunia" to the screen.

Row 7 : `return 0;`

Returning the `main()` function (A function ends when it returns)

Row 8 : `}`

Closing the `main()` function code block.

5.1 Exercise

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

6 Data Types and Variable

6.1 Data Types

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

Tabel 1: Some data types in C programming language

Data Types	Size	Range Value
Int (or signed int)	2 bytes	-32,768 to 32,767
unsigned int	2 bytes	0 to 65,535
Short int(or signed short int)	2 bytes	-32,768 to 32,767
Long(or signed short int)	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295
float	4 bytes	1.2E-38 to 3.4E+38
double	8 bytes	2.3E-308 to 1.7E+308
Long double	10 bytes	3.4E-4932 to 1.1E+4932
char(or signed char)	1 byte	-128 to 127
unsigned char	1 byte	0 to 255

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.

Tabel 2: Format Specifier

Format Specifier	Data Types
%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. These data types and its specification can be found easily on the internet.

6.1.1 Modifier

In general, a modifier is a word, phrase, or clause that modifies or describes a noun or verb in a sentence. Meanwhile, in programming languages, a modifier is a keyword used to alter the behavior or characteristics of an element in a program, such as variables, functions, or classes. We use modifiers to change the range of basic data types to fit programming needs. There are four modifiers, namely:

1. signed

`int value = -10;` (Using a negative sign for variables of type int, which are signed integers by default.)

2. unsigned

`unsigned int count = 100;` (Using a negative sign for variables of type int, which are signed integers by default.)

3. long

`long population = 7500000000;` (Using the long data type to store values larger than the int data type.)

4. short

`short temperature = 20;` (Using the short data type to save memory when we know that the values to be stored will be relatively small.)

6.2 Variable

Variables are places to store data. Declaring a variable can be done in the following ways

Listing 2: C variable declaration

```
1 DataType VariableName;
```

6.2.1 Aithmetic and Assignment Operator

Assignment operator can be use in variabel that is not a const or a variable that has -1 value. However, arithmetic operators can accept both variables with 'const' or without ('left-hand' and 'right-hand' values) The table below shows some arithmetic operators in C

Tabel 3: Arithmetic operator in C programming language

Operator	Name	Example
+	Penambahan	$x + y$
-	Pengurangan	$x - y$
*	Perkalian	$x * y$
/	Pembagian	x/y
%	Modulo	$x \% y$

Tabel di bawah menunjukkan beberapa operator penugasan.

Tabel 4: Assignment operator

Operator	Example	Similiar meaning
=	$x = 5$	$x = 5$
+=	$x += 3$	$x = x + 3$
-=	$x -= 3$	$x = x - 3$
*=	$x *= 3$	$x = x * 3$
/=	$x /= 3$	$x = x / 3$
%=	$x \% = 3$	$x = x \% 3$
&=	$x \& = 3$	$x = x \& 3$
=	$x = 3$	$x = x 3$
^=	$x \wedge = 3$	$x = x \wedge 3$
>>=	$x \gg = 3$	$x = x \gg 3$
<<=	$x \ll = 3$	$x = x \ll 3$

There are also 'abbreviations' for some assignment operators such as $x+=1$ and $x-1$, which $++$ and $--$. There are also 'abbreviations' for some assignment operators such as.

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

6.2.2 Operator Bitwise

Bitwise operators are special operators used to handle logical operations on binary numbers in bit form. Binary numbers themselves are a type of number consisting of only two digits, which are 0 and 1. If the original value used is not in binary, it will be automatically converted by the C compiler into a binary number. For example, 7 in decimal equals 0111 in binary

Tabel 5: Bitwise operator

Operator	Name	Example	Binary	Result (binary)	Result (decimal)
&	AND	10 & 12	1010 & 1100	1000	8
	OR	10 12	1010 1100	1110	14
^	XOR	10 ^12	1010 ^1100	0110	6
~	NOT	~10	~1010	0101	-11 (two complement)
<<	Left shift	10 <<1	1010 <<1	10100	20
>>	Right shift	10 >>1	1010 >>1	101	5

Notes: There are several operators in the C language. Please study them by seeking references independently

6.3 Exercise

Listing 3: Using assignment operator in a const variable

```
1 #include <stdio.h>
2 int main()
3 {
4
5     //variable declaration
6     const int x=0;
7     x=1;
8     return 0;
9 }
```

Try to compile the program in Listing3, what happened?

7 Input and Output

7.1 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 2

Contoh 7.1.1 Printing text to the screen.

Listing 4: Print text "C Programming" Ke layar

```
1  #include <stdio.h>
2  int main()
3  {
4      // Printing text inside 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.
- return 0; pernyataan di main() fungsi memberitahu program untuk keluar.

Contoh 7.1.2 Printing integer.

```
1  #include <stdio.h>
2  int main()
3  {
4      int testInteger = 5;
5      printf("Number = %d", testInteger); // <- %d format string
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 7.1.3 Real number output (float atau double)

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

$$Area = \frac{1}{2} \times Base \times Height \quad (1)$$

```
1  #include <stdio.h>
2
3  int main()
4  {
5      // variable declaration
6      float Base;
7      float Height;
8      float Area;
9      // value initialization
10     Base = 10;
11     Height = 5;
12     // calculating area
```

```

13     Area = 0.5*Base*Height;
14     // printing the text to screen
15     printf("Area = %f",Area);
16     return 0;
17 }
18
19

```

explanation

Row 6-8 Base, Height and Area are float data type which use to store triangle area data.

Row 10 dan 11 Value initialization to Base=10 and Height=5

Row 13 Calculating triangle area according to the equation 7.1

Row 15 Printing Area to the screen using printf command.

.

7.2 scanf

Function `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 successfully read.

Example 7.2.4 Calculating triangle Base dan tinggi Height yang diinputkan dari keyboard.

```

1  #include <stdio.h>
2
3  int main()
4  {
5      float Base, Height, Area;
6
7      printf("Calculate triangle area\n");
8      printf("\nInsert Base= ");
9      scanf("%f",&Base);
10     printf("\nMasukkan Height=");
11     scanf("%f",&Height);
12     Area = 0.5*Base *Height;
13     printf("Triangle Area = %.2f", Area);
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.
```

Gambar 9

Row 9 `scanf("%f",&Area);` requesting input for triangle base

Row 11 `scanf("%f",&Height);` requesting input for triangle height

Row 13 `printf("Triangle Area = %.2f", Area);`, `.2` si `%.2f` indicating that only 2 digits after the decimal point need to be printed

Example 7.2.5 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`. Because the text contains many characters, each variable is declared as an array of characters with the number of characters for `sName=20` and `sEmailAddress=30`.

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

7.3 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.

Tabel 6: Escape Sequence

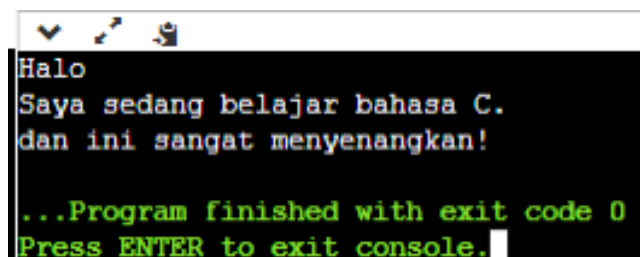
Escape sequence	Output
\a	Bell, alarm
\b	Backspace
\f	Change Page
\n	Change Row
\r	Carriage return
\t	Tab Horizontal
\v	Tab Vertikal
\'	Single Quotes
\"	Double Quotes
\?	Question Mark
\\	Backslash

Contoh 7.3.6 Change row with escape sequence \n.

```

1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Halo \nI'm learning C programming language.\nand it's so fun!");
6     return 0;
7 }
8

```



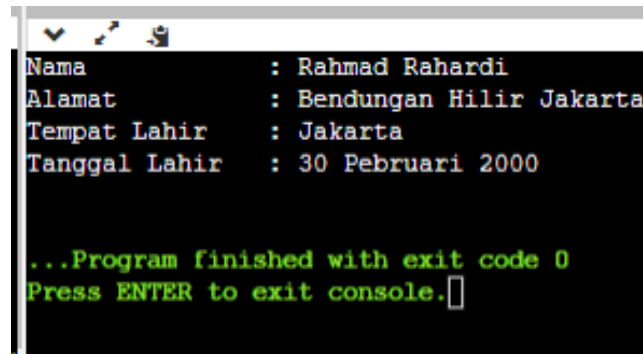
Gambar 10

Contoh 7.3.7 Using escape sequence \t to change tab.

```

1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Name \t\t: Rahmad Rahardi\n");
5     printf("Address \t\t: Bendungan Hilir Jakarta\n");
6     printf("Place of Birth \t: Jakarta\n");
7     printf("Date of Birth \t: 30 February 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.
```

Gambar 11

7.4 Exercise

1. Try to build a program to read input for the name and NRP and display it to the screen.
2. Try to build a program that asks user to enter a Celcius temperature and then convert it to Farenheit. .

Optional: Learn Git and Github. You can start learning from the following resources:

GitHub - <https://github.com>

Git - <https://git-scm.com/doc>

8 Goals

- Students are familiar with and able to use logical and comparison expressions in the C programming language.
- Students are familiar with and able to use logical and comparison expressions in the C programming language.
- Students can recognize and use while loops in the C language.
- Students can recognize and use do-while loops in the C language.
- Students can recognize and use for loops in the C language.
- Students can recognize and use for loops in the C language.
- Students can recognize and use for loops in the C language.
- Students can recognize and use strings.

9 Logical and Comparasion Expressions

9.1 Comparasion Expressions

The following are the operators used in comparison expressions.

Tabel 7: Comparasion Operator

Operator	Name	Expression Example
==	Equal To	$x == y$
!=	Not Equal To	$x != y$
>	Greater Than	$x > y$
<	Less Than	$x < y$
>=	Greater Than Equal To	$x >= y$
<=	Less Than Equal To	$x <= y$

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

9.2 Logical Expression

The following are the logical operators used on a Logical Expression

Tabel 8: Logical Expression

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

Like comparison expression, logical expression will return boolean values.

10 Branch

10.1 If Statement

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

```
// Block code before if
if (Condition)
{
    // Block of code that will be executed if the condition is true
}
// Block code after if
```

As example, look at the following code

Listing 5: If Statement Example

```
1  include <stdio.h>
2
3  int main()
4  {
5      //Variable Declaration
6      int myMoney, breadPrice;
7      myMoney = 5000;
```



```

8      breadPrice = 10000;
9
10     if (myMoney >= breadPrice)
11     {
12         printf("I can buy that bread\n");
13     }
14     printf("hehe");
15     return 0;
16 }

```

Output of the program

hehe

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

I can buy that bread
hehe

10.2 If-else Statement

Else statement is used to decide the block of code to be executed if the condition is false.

```

// Block code before if
if (Condition)
{
// Block of code that will be executed if the condition is true
} else
{
// Block of code that will be executed if the condition is false
}
// Bloc code after if-else

```

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

Listing 6: If-else example

```

1  include <stdio.h>
2
3  int main()
4  {
5      //Variable declaration
6      int myMoney, breadPrice;
7      myMoney = 5000;
8      breadPrice = 10000;
9
10     if (myMoney >= breadPrice)
11     {
12         printf("I can buy that bread\n");
13     }
14     else
15     {
16         printf("I can't buy that bread\n");
17     }

```

```

18     printf("hehe");
19     return 0;
20 }

```

Below is the output of that program

```

I can buy that bread
hehe

```

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

```

I can't buy that bread
hehe

```

10.3 Pernyataan if-else if

The `else if` statement is used to run a block of code when the condition in `if` or the previous `else if` is false.

```

// Code block before the if statement
if (Condition1)
{
    /* Code block to be executed if Condition 1
    is true */
}
else if (Condition2)
{
    /* Code block to be executed if Condition 1 is false
    and Condition 2 is true */
}
else if (Condition3)
{
    /* Code block to be executed when
    Condition 1 and Condition 2 are false, and
    Condition 3 is true */
}
...
else if (ConditionN)
{
    /* Code block to be executed when
    Condition 1 to Condition N-1 are false, and
    Condition N is true */
}
else
{
    /* Code block to be executed when
    Condition 1 to Condition N are false */
}

```

```
// Code block after the if statement
```

Below are an example of if-else statement

Listing 7: If-else example if

```
1  include <stdio.h>
2
3  int main()
4  {
5      //Variable declaration
6      int myMoney,breadPrice;
7      myMoney = 5000;
8      breadPrice = 10000;
9
10     if (myMoney>breadPrice)
11     {
12         printf("I can buy that bread\n");
13     }
14     else if(myMoney==breadPrice)
15     {
16         printf("I can buy bread, but my money will run out immediately");
17     }
18     else
19     {
20         printf("I can't buy that bread\n");
21     }
22     printf("hehe");
23     return 0;
24 }
```

Output of this program are below

```
I can't buy that bread
hehe
```

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

```
I can buy bread, but my money will run out immediately
hehe
```

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

```
I can buy that bread
hehe
```

10.4 Nested if

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

```
// Code block before the if statement
if (Condition1)
{
```

```

if (Condition2)
{
// Do something
}
else
{
// Do other thing
}
}
else
{
// Do other thing
}
}

```

Below is an example of using nested if

Listing 8: 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\n");
25         }
26         else
27         {
28             printf("I can't buy bread\n");
29         }
30     }
31     printf("hehe");
32     return 0;
33 }

```

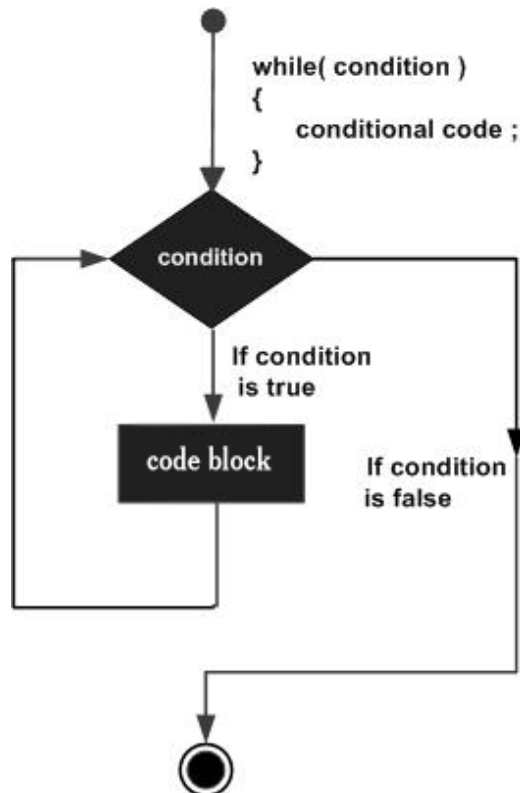
10.5 Exercise

1. 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.

11 Loop

11.1 Loop while

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



Gambar 12: Flow chart loop while

Its syntax in C programming language is as follows

```
while(Condition)
{
    // Block of code that will be repeated
}
```

As an example, look at the following code

Listing 9: While implementation example

```
1 int main()
2 {
3     int myMoney, breadPrice;
4     myMoney = 10000;
5     breadPrice = 2000;
6     while(myMoney >= breadPrice)
7     {
```

```

8     printf("Buy 1 bread, my money left %d", myMoney - breadPrice);
9     myMoney -= breadPrice;
10  }
11  printf("I don't have enough money");
12  return 0;
13 }

```

Output in program 9 are below

```

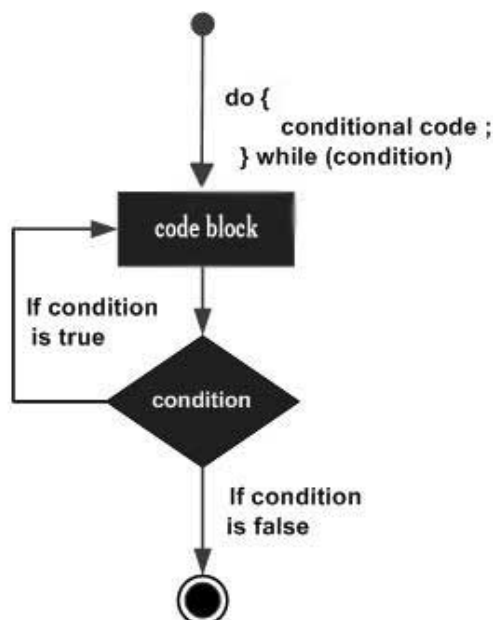
Buy 1 bread, my money left 8000
Buy 1 bread, my money left 6000
Buy 1 bread, my money left 4000
Buy 1 bread, my money left 2000
Buy 1 bread, my money left 0
I don't have enough money

```

You can see the line 9 of the code causes the variable `myMoney` to have its value subtracted by 2000 for every loop until `myMoney` is no longer greater than equal to `breadPrice`. 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.

11.2 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.



Gambar 13: Do-while statement

Its syntax in C is as follows:

```

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

```

Look at the following example.

Listing 10: Do-while implementation example

```
1 int main()
2 {
3     int myMoney, breadPrice;
4     myMoney = 10000;
5     breadPrice = 12000;
6     do{
7         printf("Buy 1 bread, my money left %d", myMoney - breadPrice);
8         myMoney -= breadPrice;
9     }while(myMoney >= breadPrice)
10    printf("Uang saya tidak cukup lagi");
11    return 0;
12 }
```

The output of the code above in Listing 10 are

```
Buy 1 bread, my money -2000
I don't have enough money
```

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

11.3 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 equal to

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

As example, look at the following code:

Listing 11: For implementation example

```
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 11 converted to its while-loop form

Listing 12: For in form of 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 }
```

Notes: In programming, the "break" keyword is used to exit a loop prematurely, while the "continue" keyword is used to skip the current iteration of a loop and proceed to the next iteration. These keywords are commonly used in loops to control their behavior and make the code more efficient. You can explore their usage further in programming resources and tutorials.!

11.4 Exercise

1. Implement a program in C that calculates the factorial of a non-negative integer entered by the user using a do-while loop. Show the results.
2. Implement programs in C language to find prime numbers between 1 and 100. Use the for loop to iterate through all numbers and the continue statement to ignore numbers that are not prime. Display all found primes.

12 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.

12.1 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 13: Accessing 1D array implementation

```
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 14 will give output

```
4
9
4
42069
```

12.2 Array 2D and Other Multidimensional Array

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}};
```

12.3 Exercise

1. Try to initialize a multidimensional array with for loop
2. Create a program to fill the data of an array by keyboard input.
3. What would happen if an array `arr` is accessed with `arr[-1]`?
4. What would happen if an array `arr` with size 5 is accessed with `arr[5]`?
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?

13 String

In general, a string is a collection of one or more characters. Specifically in the C language, a string is defined as a collection of characters terminated by a null character. `'\0'`.

For example, string "Dasar", in C programming language can be represented in a collection of character `'D'`, `'a'`, `'s'`, `'a'`, `'r'`, dan `'\0'`.

13.1 String Uses

Because a string is essentially an array of characters, creating a string data type in C follows the same approach as creating an array. Here's an example:

Listing 14: Char in string implementation

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      char foo[8] = {'b','e','l','a','j','a','r','\0'};
6      printf("Isi variabel foo adalah %s \n", foo);
7
8      return 0;
9  }
```

`'\0'` is one of requirement in order to create a string in C programming language. Every string need a "special" character to indicates it's end. This `'\0'` represented null charachter which use in C programming compiler as an indication of the end of the string

Source code implementation `scanf` to read string:

Listing 15: String with scanf implementation

```
1  #include <stdio.h>
2
3  int main() {
4      // Declare variable to store input from user
5      int age;
6      float height;
7      char name[50];
8
9      // Request user to input their age
10     printf("Enter your age: ");
11     scanf("%d", &age);
12
13     // Request user to input their height
14     printf("Enter your height (in meter): ");
15     scanf("%f", &height);
16
17     // Request user to enter their name
18     printf("Enter your name: ");
19     scanf("%s", name);
20
21     // Display user information
22     printf("Name: %s\n", name);
23     printf("Age: %d year old\n", age);
24     printf("Height: %.2f meter\n", height);
25
26     return 0;
27 }
```

Source code example gets to read string:

Listing 16: String with gets implementation

```
1  #include <stdio.h>
2
3  int main () {
4
5      char arr[100];
6      while(true)
7      {
8          gets(arr);
9
10         printf("-- %s\n", arr);
11     }
12     return 0;
13
14 }
```

String that read using scanf or gets will automatically has null character in the end.

13.2 String Functions

In the C programming language, there is a library created with the purpose of facilitating users in string manipulation. This library is stored in `<string.h>`, therefore, to access this library, an additional preprocessor directive is required, which is::

```
1 #include <string.h>
```

Learn other function in www.cplusplus.com.

13.3 Exercise

1. Create a program in C programming language that takes 2 string from the user input and decide whether those 2 string are an anagram (contains the same characters even in different order)
2. Explain the difference between string that is declared as an array of character (char array) and a string that is declared as a string data types (string literal). Explain example of using both

14 Tujuan

- Mahasiswa mengerti cara membuat dan memanggil fungsi pada bahasa pemrograman C.
- Mahasiswa mampu menggunakan passing parameter by value dan by reference pada bahasa pemrograman C.
- Mahasiswa mampu mengerti dan mengaplikasikan konsep rekursi pada bahasa pemrograman C.

15 Fungsi

Fungsi adalah sebuah kumpulan statement untuk melakukan tugas spesifik, yang bisa membutuhkan input ataupun tidak, untuk menghasilkan output yang sesuai. Keuntungan menggunakan fungsi pada bahasa pemrograman C adalah:

- Beberapa cuplikan kode dapat digunakan kembali saat menggunakan fungsi. Kita dapat memanggil fungsi C berapa kali pun dalam suatu program dan dari mana saja dalam suatu program.
- Program C yang besar dapat dibagi ke dalam beberapa fungsi sehingga dapat dengan mudah untuk dilacak.

15.1 Function Declaration

Setiap program C mempunyai minimal satu fungsi, yaitu fungsi main(). Anda juga dapat mendefinisikan fungsi selain main() Syntax :

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

- Return Type.
Tipe data yang harus dikembalikan suatu fungsi.
- function_name.
Nama fungsi

- parameters list.
Parameter dari fungsi.
- Function body.
Kumpulan statemen yang mendefinisikan apa yang dilakukan oleh fungsi.
- return something;
merupakan statement untuk mengembalikan nilai dari fungsi. Untuk fungsi yang tidak mengembalikan nilai, dapat digunakan return_type void. Untuk keluar dari fungsi itu hanya perlu menggunakan statement return

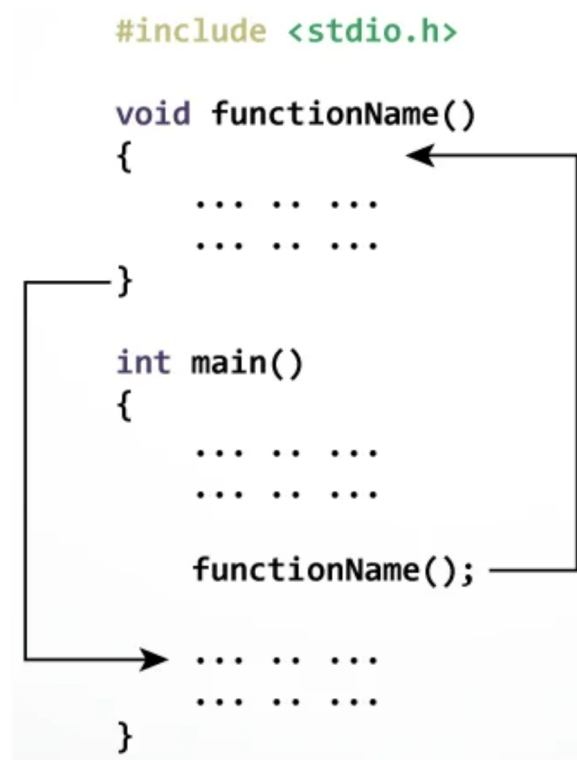
Contoh

```

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

```

15.2 Memanggil Fungsi



Gambar 14

```

1 #include <stdio.h>
2 // Mendeklarasikan fungsi luasSegitiga
3 // Parameter input Alas , dan Tinggi
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     // Memanggil Fungsi TriangleArea
15     L=TriangleArea(Bs,Hg);
16     printf("Area = %f",L);
17     return 0;
18 }

```

1. Baris 5-10:Mendefinisikan fungsi TriangleArea dengan

- 2 paramater input/masukan: input Base dan Height dengan tipe data float.
- Output bernilai tunggal dengan tipe data float.

15.3 Fungsi dengan Argumen

15.3.1 Argumen

Jika suatu fungsi diharapkan untuk menggunakan argumen, maka variabel sebagai parameter yang menerima nilai dari argumen tersebut harus di dedeklarasikan terlebih dahulu.

1. Parameter :

- Parameter adalah variabel dalam fungsi untuk merujuk ke salah satu bagian dari data yang diberikan sebagai input ke fungsi.
- Data ini disebut argumen.

2. Formal Parameter:

- Parameter yang Ditulis dalam Definisi Fungsi Disebut "Parameter Formal".
- Parameter formal selalu variabel, sedangkan parameter aktual tidak harus variabel.

3. Actual Parameter:

- Parameter yang Ditulis ketika memanggil fungsi.
- Dapat berupa angka, ekspresi, atau bahkan panggilan fungsi.

```

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

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

```

formal arguments
actual arguments

Gambar 15

15.4 Parameter Passing

Passing parameter merupakan aktivitas menyalurkan nilai pada parameter saat memanggil fungsi. Pada umumnya, dikenal dua macam passing parameter yaitu:

- Pass parameter by value, yaitu menyalurkan **nilai** dari tiap parameter yang diberikan.
- Pass parameter by reference, yaitu menyalurkan **alamat** dari tiap parameter yang diberikan.

15.4.1 Passing Parameter by Value

Listing 17: Passing by Value

```

1 #include <cstdio>
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 }

```

Perhatikan potongan kode pada Listing 17. Baris 3-6 dari kode tersebut adalah operasi untuk menukar nilai dari 2 variabel. Namun, apabila program tersebut dijalankan, maka akan muncul output

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

Nilai dari a dan b tidak bertukar. Untuk passing parameter by value, apapun yang dilakukan pada function body tidak akan berpengaruh pada parameter yang "dipassingkan". Nilai dari parameter aktual akan diassign pada parameter formal.

15.4.2 Passing Parameter by Reference

Perhatikan baris 2 pada potongan kode berikut:

Listing 18: 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 }
```

Apabila program tersebut dijalankan, maka akan muncul output

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

Ketika fungsi `swapAndReturnTheSum(a,b)` dipanggil, alamat memori variabel a dan b "dipassingkan" pada fungsinya. Sehingga pada potongan kode di baris 4-6, x dan y akan mengacu pada memori parameter aktual yang dimasukkan di baris ke 13. Ketika melakukan passing by reference, kita tidak bisa memanggil fungsi dengan parameter yang tidak memiliki alamat memori. Sebagai contoh `tukarDanKembalikansumnya(1,2)` tidak bisa dilakukan karena angka 1 dan 2 bukan variabel dan tidak memiliki alamat memori.

15.5 Tugas Pendahuluan

1. Buatlah fungsi yang dapat menerima 2 buah bilangan bulat a dan b kemudian mengembalikan nilai dari a^b
2. Masalah-masalah apa yang akan lebih mudah diselesaikan dengan menggunakan fungsi?

16 Rekursi

Rekursi merujuk kepada definisi suatu hal yang dilakukan secara berulang-ulang.

Rekursi adalah ketika suatu fungsi dalam function bodynya memanggil fungsi itu sendiri. Sebagai contoh, perhatikan potongan kode berikut:

Listing 19: Factorial dengan rekursi

```
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 $n \times factorial(n-1)$, kemudian $factorial(n-1)$ akan mengembalikan $(n-1) \times factorial(n-1-1)$. Akhirnya menjadi seperti ini:

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

16.1 Tugas Pendahuluan

1. Diberikan sebuah baris bilangan 1, 5, 14, 30, ... dst. Buatlah sebuah program yang mengimplementasikan fungsi rekursif untuk menentukan bilangan ke-n dari pola tersebut.

17 Tujuan

- Mahasiswa mengerti tentang konsep pointer pada bahasa pemrograman C.
- Mahasiswa mengerti cara membuat dan memanggil struct pada bahasa pemrograman C.
- Mahasiswa mengerti tentang algoritma sorting pada bahasa pemrograman C.
- Mahasiswa mengerti tentang algoritma searching pada bahasa pemrograman C.
- Mahasiswa mampu mengaplikasikan konsep algoritma searching dan sorting pada bahasa pemrograman C.

18 Pointer

18.1 Alamat Memori

Setiap variabel, fungsi, struct, ataupun objek lain yang dibuat dalam program mempunyai lokasi masing-masing pada memori. Alokasi setiap variabel disimpan dalam alamat memori tertentu.

Jika terdapat variabel `var` di program Anda, `&var` akan memberi alamatnya di memori.

```
1  int var = 5;
2  printf("%d\n", var);
3  printf("%p\n", &var);
```

Catatan: Output bisa berbeda-beda di tiap eksekusi.

18.2 Pengenalan Pointer

Pointer (variabel penunjuk) adalah variabel khusus yang digunakan untuk menyimpan alamat, bukan nilai.

Deklarasi variabel pointer menggunakan operator `*` di antara tipe data dan nama variabelnya.

```
1  #include <stdio.h>
2  int main()
3  {
4      int* p; // atau
5      int * p2;
6      return 0;
7  }
```

18.3 Cara Kerja Pointer

Berikut adalah cara kerja dari pointer.

Listing 20: Contoh Program Pointer

```
1  #include <stdio.h>
2  int main()
3  {
4      int* pc, c;
5
6      c = 22;
7      printf("Address of c: %p\n", &c);
8      printf("Value of c: %d\n\n", c); // 22
9
10     pc = &c;
11     printf("Address of pointer pc: %p\n", pc);
12     printf("Content of pointer pc: %d\n\n", *pc); // 22
13
14     c = 11;
15     printf("Address of pointer pc: %p\n", pc);
16     printf("Content of pointer pc: %d\n\n", *pc); // 11
17
18     *pc = 2;
19     printf("Address of c: %p\n", &c);
```

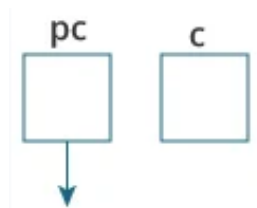
```

20     printf("Value of c: %d\n\n", c); // 2
21     return 0;
22 }

```

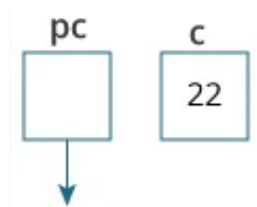
Penjelasan Program:

1. `int* pc, c;`



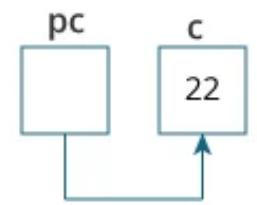
Gambar 16

2. `c = 22;`



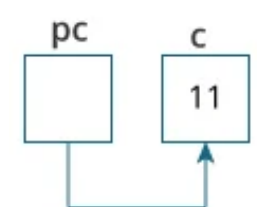
Gambar 17

3. `pc = &c;`



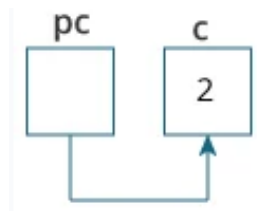
Gambar 18

4. `c = 11;`



Gambar 19

5. `*pc = 2;`



Gambar 20

18.4 Double Pointer

Variabel pointer juga dapat menunjuk variabel pointer lainnya. Hal ini disebut dengan double pointer (pointer to pointer). Untuk mendeklarasikan variabel double pointer, digunakan dua simbol *. Kegunaan paling umum dari variabel double pointer adalah untuk membuat array dua dimensi secara dinamis.

```
1 int **dbPtr;
```

Variabel dbPtr di atas menyimpan alamat memori dari variabel pointer lainnya.

Berikut contohnya

Listing 21: Contoh Double Pointer

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     int var = 23;
6     int *ptr = &var;
7     int **dbPtr = &ptr;
8
9     printf("%d\n", **dbPtr);
10
11     return 0;
12 }
  
```

18.5 Tugas Pendahuluan

1. Bagaimana cara mendeklarasikan pointer ke array multidimensi?
2. Buatlah program dalam bahasa C atau C++ yang mengimplementasikan fungsi void printMatrix(int **matrix, int rows, int cols) untuk mencetak matriks 2D menggunakan pointer ke pointer. Lalu, dalam fungsi main, buatlah matriks 2D dan panggil fungsi printMatrix untuk mencetak matriks tersebut.

19 Struct

Dalam pemrograman C, struct (atau struktur) adalah kumpulan variabel (bisa dari tipe berbeda) di bawah satu nama. Tidak seperti array yang hanya dapat menyimpan elemen dengan tipe data sama, struct dapat mengelompokkan elemen dengan tipe data yang berbeda-beda.

19.1 Deklarasi Struct

Seperti variabel, struct harus dideklarasikan terlebih dahulu sebelum bisa digunakan. Pendeklarasian struct menggunakan sintaks sebagai berikut.

```
1 struct <nama_struct> {  
2     <type_data_member> <nama_member>;  
3     <type_data_member> <nama_member>;  
4     <type_data_member> <nama_member>;  
5     .  
6     .  
7     .  
8 };
```

Berikut adalah contoh deklarasi struct berdasarkan kasus Mahasiswa.

```
1 struct Mahasiswa  
2 {  
3     char *name;  
4     char *address;  
5     int age;  
6 };
```

Catatan: Menggunakan pointer * untuk data string

Setelah dideklarasikan, sebuah struct akan menjadi tipe data baru. Maka dalam kasus ini, struct Mahasiswa di sini menjadi tipe data baru dengan member-member berupa nama, address, dan age. Untuk membuat variabel dengan tipe data struct, dilakukan dengan sintaks berikut.

```
1 struct <nama_struct> <nama_variabel>;
```

Contoh:

```
1 struct Mahasiswa mhs1;  
2 struct Mahasiswa mhs2;
```

Contoh di atas menunjukkan terdapat dua variabel mhs1 dan mhs2 bertipe struct Mahasiswa.

19.2 Akses Member Struct

Bagaimana cara untuk mengakses member dari variabel struct yang telah dibuat?

Untuk mengakses member-member dari struct, digunakan operator dot (.) setelah nama variabelnya.

```
1 <nama_variabel>.<member_struct>
```

Contoh:

```
1 mhs1.age = 20;  
2 mhs1.nama = iqbal;  
3  
4 mhs2.nama = fatur;  
5 mhs2.age = 21;
```

19.3 Tugas Pendahuluan

1. Buatlah sebuah struct yang merepresentasikan informasi tentang seorang mahasiswa, yang memiliki nama, nim, dan nilai IPK. Kemudian, buatlah program untuk menginput data mahasiswa, menampilkan data mahasiswa, dan menghitung rata-rata IPK dari sejumlah mahasiswa.
2. Anda diberikan struct yang merepresentasikan titik dalam sistem koordinat dua dimensi (x, y). Buatlah sebuah program C untuk menghitung jarak antara dua titik yang diinputkan oleh pengguna menggunakan rumus jarak Euclidean.

20 Algoritma Sorting

Sorting merupakan suatu proses penyortiran atau pengurutan sebuah data.

Terdapat 2 macam pengurutan data pada sorting yaitu :

1. Berdasarkan ascending (kecil ke besar).
2. Berdasarkan Descending (besar ke kecil).

20.1 Bubble Sort

Bubble sort merupakan algoritma pengurutan yang membandingkan dua data yang berdekatan dan menukarnya sampai tidak dalam urutan yang diinginkan. Bubble sort menggunakan teknik iterasi. Iterasi merupakan proses melakukan perulangan sebanyak data yang diketahui. Intinya pada iterasi melakukan perbandingan antara dua data.



Gambar 21

Listing 22: Implementasi Bubble Sort

```
1 void swap ( int * xp , int * yp ) {  
2     int temp = *xp;  
3     *xp = *yp;  
4     *yp = temp;  
5 }
```

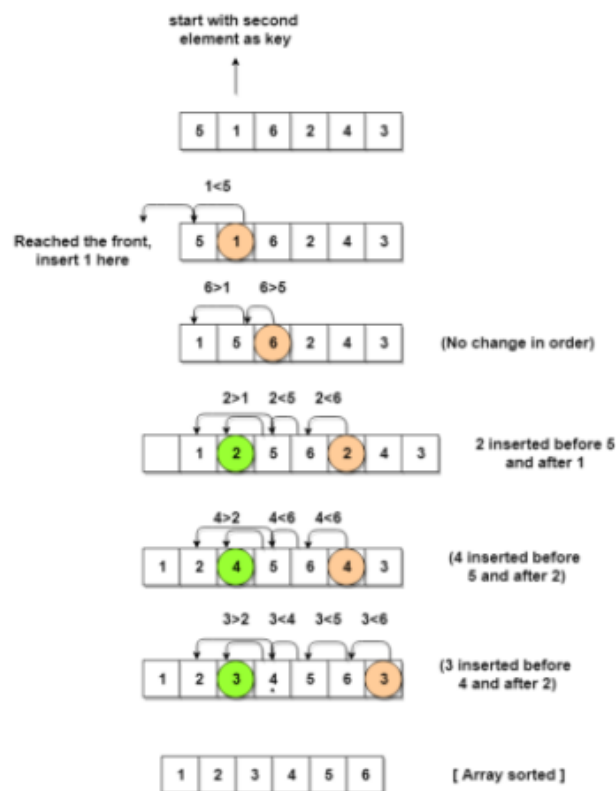
```

6
7 void bubbleSort(int arr[], int n) {
8     int i, j, swapped;           // dioptimasi dengan bool 'swapped':
9     for (i = 0; i < n-1; i++) {
10         swapped = 0;
11         for (j = 0; j < n-i-1; j++) {
12             if (arr[j] > arr[j+1]) {
13                 swap(&arr[j], &arr[j+1]);
14                 swapped = 1;
15             }
16         }
17         if (swapped == 0)
18             break;
19     }
20 }

```

20.2 Insertion Sort

Insertion sort merupakan teknik sorting dengan cara menyisipkan atau memasukan setiap elemen secara berulang berulang. Konsep insertion sort bisa diibaratkan sebuah kartu.



Gambar 22

Listing 23: Implementasi Insertion Sort

```

1 void insertionSort(int arr[], int n) {
2     int i, key, j;
3     for (i = 1; i < n; i++) {
4         key = arr[i];

```

```

5      j = i-1;
6
7      while (j >= 0 && arr[j] > key) {
8          arr[j+1] = arr[j];
9          j = j-1;
10     }
11     arr[j+1] = key;
12 }
13 }

```

Catatan: Terdapat berbagai algoritma sorting lain. Pelajari secara mandiri

20.3 Tugas Pendahuluan

1. Urutkan array berikut menggunakan algoritma Bubble Sort:

Array: [5, 2, 9, 1, 5, 6]

2. Hitung kompleksitas waktu (Big O) dari algoritma Insertion Sort saat mengurutkan sebuah array dengan panjang n , dan jelaskan bagaimana kompleksitas ini dihitung.

21 Algoritma Searching

Searching merupakan proses pencarian sebuah data yang diinginkan.

21.1 Linear Search

Linear Search bekerja dengan melakukan pengecekan kepada semua elemen yang ada.

Secara garis besar, cara kerja Linear Search adalah:

1. Memeriksa item satu per satu.
2. Apabila ditemukan, maka “ketemu”.
3. Jika sampai akhir belum ditemukan, maka item yang dicari tidak ada.

Listing 24: Implementasi Linear Search

```

14 int linearSearch(int arr[], int n, int item) {
15     int i;
16     for(i = 0; i < n; ++i) {
17         if(item == arr[i])
18             return 1;
19     }
20     return -1;
21 }

```


21.2 Binary Search

Binary Search adalah teknik pencarian di mana untuk setiap iterasinya kita membagi space pencarian menjadi hanya setengah dari space pencarian awal hingga kita menemukan yang kita cari.

Listing 25: Implementasi Binary Search

```
22 bool f(int k, int a, int b, int n) {
23     return ((k/a) * (b/a) >= n);
24 }
25
26 int binser(int a, int b, int n) {
27     int l = 1;
28     int r = 100000;
29     while (r - l > 1) {
30         int mid = (l + r) >> 1;
31         bool can = f(mid);
32         if (can)
33             r = mid;
34         else
35             l = mid + 1;
36     }
37     if (can(l))
38         return l;
39     else
40         return r;
41 }
```

Catatan: Terdapat berbagai algoritma searching lain. Pelajari secara mandiri

21.3 Tugas Pendahuluan

1. Anda memiliki daftar nama berikut: ["Alice", "Bob", "Charlie", "David", "Eve", "Frank"]. Gunakan algoritma binary search untuk mencari apakah nama "Eve" ada dalam daftar ini. Jika ya, berapa langkah yang dibutuhkan?
2. Jelaskan perbedaan antara pencarian linear (sequential search) dan pencarian biner (binary search). Kapan Anda akan memilih salah satu metode ini daripada yang lain?