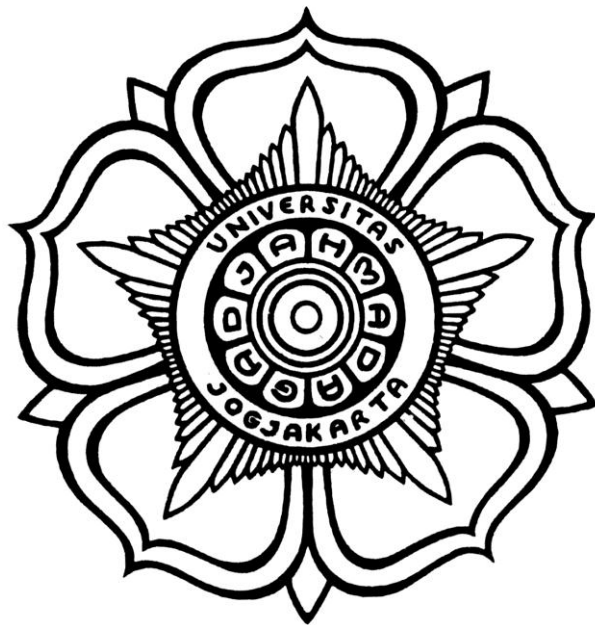


**PRACTICAL HANDBOOK**

# **PROGRAMMING I**

**(MII1202)**



**BASIC LABORATORY OF COMPUTER**  
**Department of Computer Science and Electronics**  
**Faculty of Mathematics and Natural Sciences**  
**Universitas Gadjah Mada**  
**Yogyakarta**  
**2017**

## OFFICIAL RULES

1. The students should enter the class according to schedule agreement, if more than 15 minutes late, there is no permission to enter the class.
2. The students who has been more than 3 times absence-without-notice are not permitted to participate in final examination.
3. There is no substitute for midterm or final examination, except if there is an agreement from lectures.
4. The students must wear appropriate clothes:
  - **Boys** :
    - Long or short sleeve shirt
    - Trousers
    - Must wear shoes
    - Collarless T-shirt or sleeveless shirt not allowed
  - **Girls** :
    - Long or short sleeve shirt (not tight and transparent)
    - Skirts OR trousers (not tight and transparent)
    - Must wear shoes
    - Collarless T-shirt or sleeveless shirt not allowed
5. No food, no drink, or use of cigarette in any form is allowed in the class.
6. Bags and other equipment should be placed in a provided place (the students only allowed to carry valuable belongings, e.g.: mobile phone, wallet, laptop, and tools related to activities).
7. The students are responsible for their own belongings (the laboratory will not be responsible for loss of data, damage, loss of personal items).
8. Turn off cell phones. If you need to use it, please take it to the hallway.
9. The students must keep the room clean and also conducive learning conditions.
10. The students should maintain professional and courteous communication. Electronic devices should be used on a professional level. No obnoxious or belligerent behaviour will be tolerated.
11. Software may be installed by Computer Labs staff only. Do not install any software on your own. Files not put on by Computer Labs staff will be routinely removed.
12. Please operate the equipment with respect and care.
13. The students should fill the attendance list form and write seats/computers number used during the class.

14. For any hardware, software, problems, please contact an aide.
15. The lab aides are here to help when they can and to maintain the labs' operation. However, the lab aides are not here to do your work for you. The lab aides will refer students to their instructors at the lab aides discretion.
16. The students should turn off the computers and place chairs, tables, and other equipment into a good form after finishing class.
17. Any failure to follow these lab rules may result in the loss of your lab privileges.

**BASIC LABORATORY OF COMPUTER  
DEPARTMENT OF COMPUTER SCIENCE AND ELECTRONICS  
FACULTY OF MATHEMATICS AND NATURAL SCIENCES  
UNIVERSITAS GADJAH MADA**

## **PREFACE**

Practical activities in the laboratory is an instructional activities to help students understand the theory courses are guided by an instructor and supervised by a lecturer. This practicum will focus it has standard guidelines, which are able to handle students and instructors to manage class laboratory.

This practical guide book is a revision of a previous book, with additional material, exercises and assignments are adjusted.

Hopefully, this book useful for learning the progress of computer science and improving the quality of learning.

**BASIC LABORATORY OF COMPUTER  
DEPARTMENT OF COMPUTER SCIENCE AND ELECTRONICS  
FACULTY OF MATHEMATICS AND NATURAL SCIENCES  
UNIVERSITAS GADJAH MADA**

## TABLE OF CONTENTS

<b>CHAPTER I</b>	<b>INTRODUCTION.....</b>	<b>1</b>
	1.1. Learning Objective.....	1
	1.2. Theory .....	1
	1.3. Activity.....	2
	1.4. Exercise.....	3
<b>CHAPTER II</b>	<b>INTRODUCTION TO C ++ .....</b>	<b>4</b>
	2.1. Learning Objectives .....	4
	2.2 Theory .....	4
	2.3. Activity.....	20
	2.4. Exercise .....	20
<b>CHAPTER III</b>	<b>SEQUENCE AND CONTROL STATEMENT.....</b>	<b>21</b>
	3.1. Learning Objectives .....	21
	3.2. Theory .....	21
	3.3. Activity.....	31
	3.4. Exercise.....	32
<b>CHAPTER IV</b>	<b>RECURRENCE .....</b>	<b>33</b>
	4.1. Learning Objectives .....	33
	4.2. Theory .....	33
	4.3. Activities .....	35
	4.4. Exercise.....	40
<b>CHAPTER V</b>	<b>ARRAY DATA TYPE .....</b>	<b>41</b>
	5.1. Learning Objectives .....	41
	5.2. Teory .....	41
	5.3. Activity.....	48
	5.4. Exercise.....	48
<b>CHAPTER VI</b>	<b>STRUCT DATA TYPE .....</b>	<b>49</b>
	6.1. Learning Objectives .....	49
	6.2. Teory .....	49
	6.3. Activities .....	51
	6.4. Exercise.....	53

<b>CHAPTER VII</b>	<b>SUBPROGRAM AND FUNCTION.....</b>	<b>54</b>
	7.1. Learning Objectives .....	54
	7.2. Theory .....	54
	7.3. Activity.....	60
	7.4. Exercise.....	60
<b>CHAPTER VIII</b>	<b>SORTING ALGORITHM &amp; SEARCH ALGORITHM.....</b>	<b>61</b>
	8.1. Learning Objectives .....	61
	8.2. Theory .....	61
	8.3. Activity.....	74
	8.4. Exercise.....	74
<b>CHAPTER IX</b>	<b>POINTER .....</b>	<b>75</b>
	9.1 Learning Objectives .....	75
	9.2 Theory .....	75

# CHAPTER I

## INTRODUCTION

### 1.1. Learning Objective

- Students know the definition of algorithms and programming
- Students know the basic concept of mapping algorithms into algorithmic language.
- Students are able to solve the case or the solution for a given problem

### 1.2. Theory

#### a. Background

Algorithm means solution, the term of solution in the programming language is solving a problem that must be solved using a computer. What steps are needed to solve the encountered problems . Therefore, the algorithm is the core of a programming, the algorithm must be made successively so that the computer understands and can execute the created programs correctly.

#### b. Example

Real example to portray the solution of problems that will be solved, for example, a solution to make instant noodles. The steps to create instant fried noodles are :

1. Boil water to boiled and then enter the noodles for 3 minutes.
2. Mix seasoning, seasoning oil, soy sauce, and chili powder into the dish.
3. Drain the noodles, then mix the noodles into the spice mixture on a plate, mix well
4. Fried noodles ready to be served.

As humans, we've certainly understand the steps in the making of instant noodles, but the problem in programming is how we can make computers understand step by step to produce what any results we want. In order to be able run on a computer, then the steps of preferred solution should use reasonable language for computers which are packaged in the form of a computer program.



Algorithmic language (pseudo-code) is an intermediate language between humans and computers. Pseudo-code is designed to simplify the algorithm by human logic transformed into any programming language that understood by computers. There are many programming languages that recognized by the computer, for example, Pascal, Java, PHP, C #, C ++, and so on.

Programs are algorithms coupled with data structures. The data structure is a data storage structure required by program on the computer. If humans have a brain to store the data, the computer also needs a place to store the required data. This is due to the data storage on a computer has limited capabilities than the human brain, therefore require an order or structure so that the stored data can be accessed easily..

Learning for programming means learning to make problem-solving strategies or create a solution. While programming language is a tool for learning the programming. In this lab, we will learn to make program using C++ programming language.

### 1.3. Activity

Using the previous example about the making of instant fried noodles. We can make the order in the form of good and structured algorithms..

1. **Task 1 : Declarations**, is a step to declare a place that used to make the noodles.

- 1) **Step 1** : Declare an empty place used as a place to boil the noodles.
- 2) **Step 2** : Declaring a place to put the noodles
- 3) **Step 3** : Sample code in a algorithmic language at this declaration step is as follows:

```
mie : integer
bumbu : integer
air_mendidih : integer
```

2. **Task 2 : Initialization**, this step is preparing that process done to resolve the problem.

- 1) **Step 1** : Prepare the amount of noodles, water and spice in a balance.
- 2) **Step 2** : Sample code with algorithmic language is as follows:

```
mie ← 1
air_mendidih ← 1
bumbu ← 1
```





3. **Task 3 : The process of problem solving**, is the step for problem solving to meet the goal of an algorithm we made.

1) **Step 1 : Boil noodles**

$\text{mie} \leftarrow \text{mie} + \text{air mendidih}$

2) **Step 2 : Mixing a flavor**

$\text{bumbu} \leftarrow \text{bumbu} + \text{minyak\_bumbu} + \text{kecap\_manis} + \text{bubuk\_cabe}$

4. **Task 4 : Finalization**, is a clean-up stage or final stage for example : removing the allocation of places that no longer needed, or serve the fried noodles.

1) **Step 1 : Dispose the rest of the cooking water that is not used anymore.**

$\text{air\_mendidih} \leftarrow 0$

2) **Step 2 : Serving the results of fried noodles.**

Output (“ Mie telah matang dan siap disantap”)

#### 1.4. Exercise

1. Create an algorithm to register in UGM
2. Create an algorithm for adding two numbers
3. Make an algorithm to determine the number counted as odd or even
4. Create an algorithm to calculate the area of a circle



## CHAPTER II

### INTRODUCTION TO C ++

#### 2.1. Learning Objectives

- Students know the basics of C++ programming language.
- Students can transform the algorithmic language into C++ programming language.
- Students are able to create programs by starting from a simple case using the C++ programming language.
- Students know the basic types integer, real, character, string, and boolean.
- Students understand how to implement the operation of the basic data types in C++ programming language.
- Students know the definitions and the various operators.
- Students are able to implement these operators in a given case study.

#### 2.2 Theory

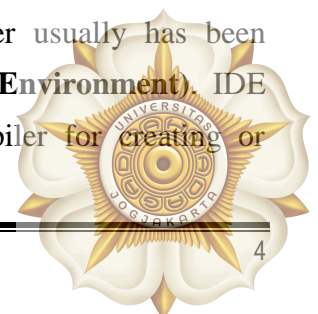
##### Background

Armed with the basic concepts of algorithms and programming, so we can further learn programming language that we will use in this time to learn the C++ programming language. The solutions of the problems that we face need to be mapped into the algorithmic language to make it more easily understood in the form of the programming language used. Then algorithmic language is mapped into C++ in accordance with the structure of C++ programming language.

##### 2.2.1 C++ Programming

The C++ programming language was the development of the C programming language as its predecessor. The term of C++ is often known as "C with Classes" because the programming language C++ has supported object-oriented programming as well as Java programming language. All the existing library on the C programming language has also been included in the C++ programming language

The *compiler* is a software that used to change the program code (source code) into machine language (**binary file**) so it can be executed by the computer. The program will be able compiled if the program contains no errors at all in its rules (syntax error). Examples for popular C/C++ compiler are MinGW and GCC, the compiler usually has been packaged/bundled with its IDE software (**Integrated Development Environment**). IDE software is an integrated interface with facilities including a compiler for creating or



development of the program. The popular and opensource C/C++ IDE Software that are Codeblock, Dev C++, and Sublime (for Mac OS)

The C++ Programming language has case sensitive characteristic, which means the compiler distinguish uppercase and lowercase letters, for example, if we write **printf** and **Printf** in the C language, the C compiler will consider both of the texts are in different meanings. In practical of C++ programming language today, we are going to use Dev C++ IDE which also as the *open source* compiler program..

The parts that support the making of a program created by the C++ language programming, there are:

- **Comment**, is part of the program code which not executed by the compiler. Comments are considered importantly to clarify the program to be more easily understood and provide information on specific parts of program code. By commenting, our program can be read easily by other to be developed further. Usually without comment, people will be hard to understand the flow of the program code that been made, therefore the comments are needed in order to make our code more informative.
- **Identifier**, the name given by the programmer (the person who make the program). Naming an identifier can be used in the program name, the name of the function, or other objects that involved in the programming language, such as variable names, constants which will be discussed further.
- **Keywords** are specific words that contain special meaning contained in the programming language. In C++ programming language, which called as keyword are asm,instance, class, delete, friend, inline, new, operator, private, protected, public, template, this, virtual, etc.. The words are considered as keywords according to the standard of a programming language should not be used as an **identifier** name.
- **Library of function**, in contrast with the keyword, function library is a library that contains the functions provided by the C ++ language in the header files or the library. The functions are used to perform a particular tasks. Functions are grouped according to the type also characteristic and stored in a file with extension **.h** . For example, a function **cout** which is stored in the library file **iostream.h** , used to print on the screen. Default functions of the C language, for example, **printf** from the **stdio.h**



library can also be used in C++ if we write **#include <stdio.h>** at the beginning of the program.

### Program structure

```
// my first program in C++                Hello World
#include <iostream.h>
using namespace std;
int main () {
    cout << "Hello World!";
    /* Ini juga baris komentar */
    return 0;
}
```

- **// my first program in C++** , is a comment line that begins with two oblique sign (/ /) or are enclosed in / \*\* / and has no effect on the program. In this case, this comment line is used to describe a piece of code ever made.
- **#include <iostream.h>** , preceded by a sharp mark (#) or, this line is a preprocessor line. In this case, **#include <iostream>** states to include iostream standar file. This particular file includes the declarations of the basic standard input-output library in C ++ language. Functions that commonly are used by beginner programmers of the iostream.h library include: cin, cout, system ("pause").
- **using namespace std;**, this line tells the compiler that the program is written using standard C ++ library. Sometimes with the **using namespace std;** programmer does not need to write the (.h) on writing the standard C++ library.
- **int main ()** , this line is a main ()function. This line is at the core of the program that may include variables, assignment statements, or commands. This line consists of a series of source code that begins with the opening brace { and closing brace } and mark {} means showing where the main ()function begins and ends or the so-called **code blocks**. So when a function is called, the content where enclosed by a block of code to be executed.
- **cout << "Hello World!";** this line is a statement of C++. A statement is a simple expression that can produce several effects. **Cout** represents the command of standard output in C++, **cout** is declared in the iostream standard file within the std namespace. So this line of code used to display the phrase "hello world".



*Notice that the statement ends with a semicolon character (;). This character is used to mark the end of the statement and must be written at the end of all expression statements in all C ++ programs .*

- **Return 0;** return instruction causes the main ()function terminated and returns the code that follows the instructions, in this case is 0. This is the most commonly used to end the program..

## Implementation C on Dev C++

### 1. Task 1 : Create a new project in devc ++ in Windows OS

#### 1) Step 1 : In C ++ application, select File> New Project > New Source File

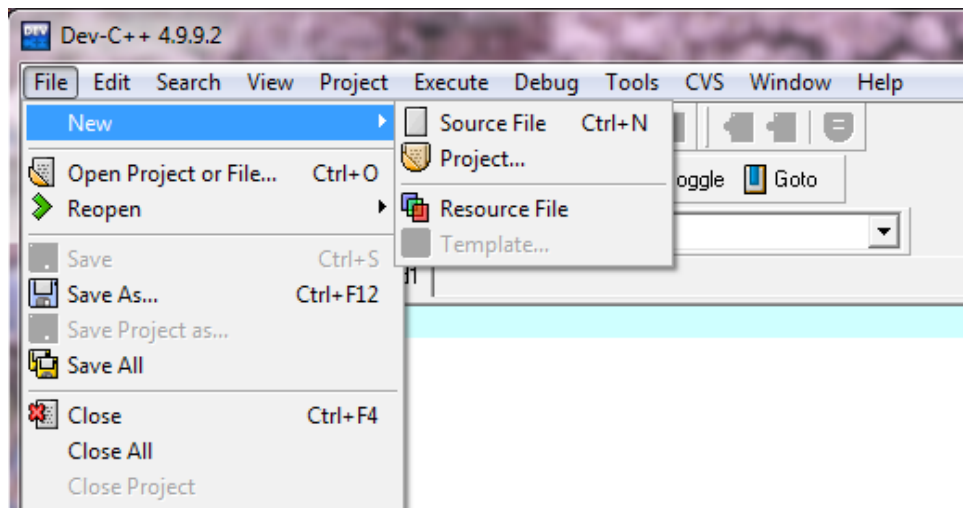


Figure 2.1 Display of new projects

#### 2) Step 2 : Type the code in the work area

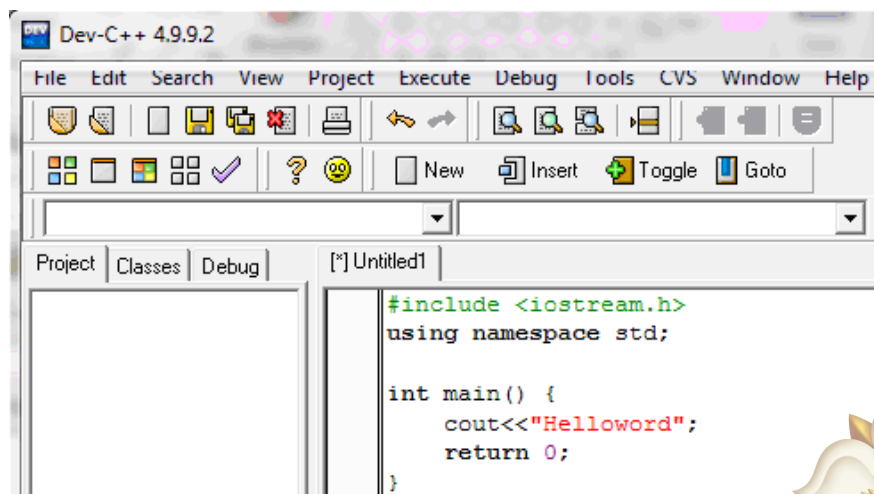


Figure 2.2 Display of source code



## 2. Task 2 : Save the new project

- 1) Step 1 : Select File Menu> Save As >
- 2) Step 2 : Select the storage directory and file name

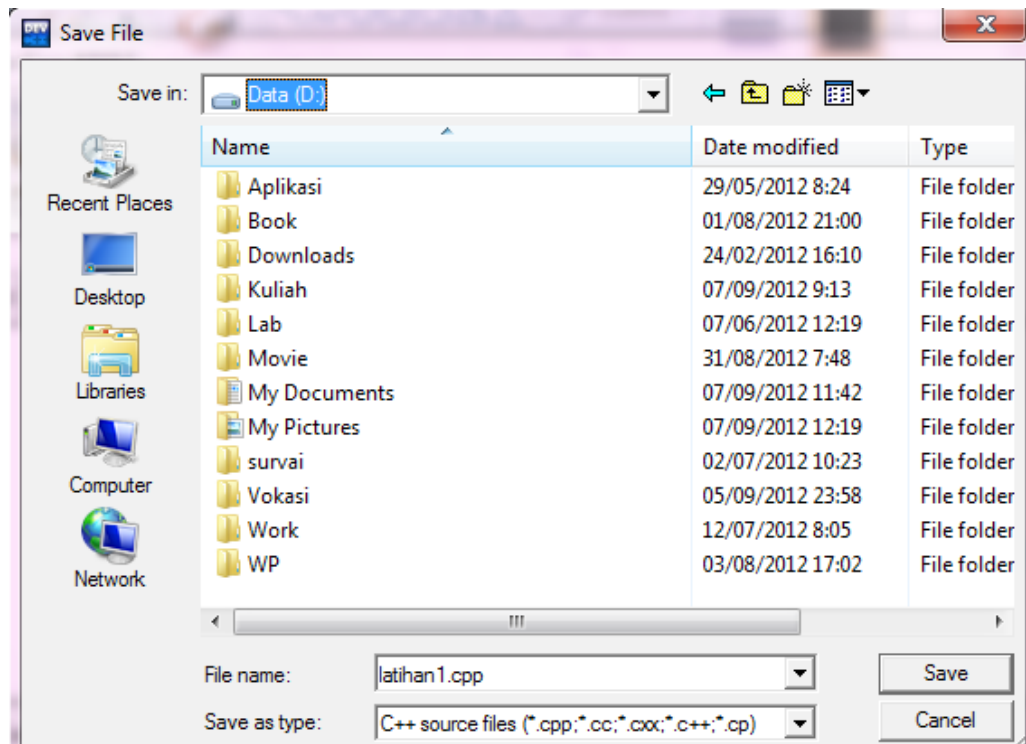


Figure 2.3 Display of the Save As

## 3. Task 3 : Compilation and Execution of program

- 1) Step 1 : Select the Execute menu> Compile

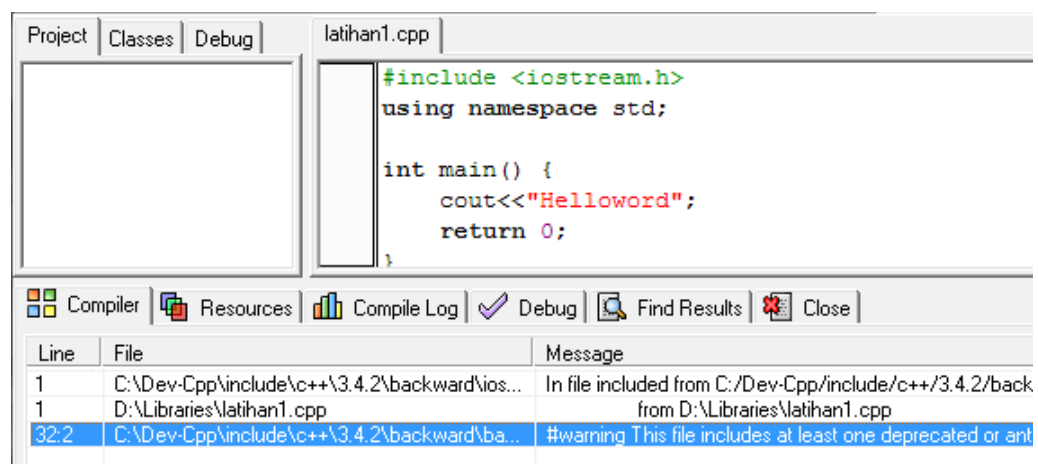


Figure 2.4 Display of compile program



## 2) Step 2 : Select the Execute menu > Run

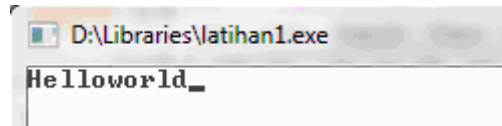


Figure 2.5 Display of the running program

## 2. Task 2 : Create a new project in a Linux OS

Programming in C++ on the Linux operating system using g++ software%% GNU C ++. This compiler can be automatically installed when installing the Linux operating system if we selected the option to install g++.

Compilation by using g++ is done in the Linux console (terminal). How that is done is, at the first please type C ++ source file in a text editor and then saved as a file with extension. Cpp,. Cp, cxx, c ++, cp, or cc (example: **coba1.cpp**). Then place it in the folder home / <username> user>. Type these command

```
g++ <path/nama file.cpp> -o <file output>
```

*Example :* **g++ coba1 -o keluaran** (then press enter).

**The file path** is not required when we first go into the folder where the source file is stored. Output file may not be given, however by default, g++ will create the output file named **a.out**. In this example if there is no error then the compilation will produce (output) binary file in the name of **keluaran**.

Once the compilation is done, check whether the compilation was successful by looking at the results of compilation messages. If there are any error messages then the output file has not been established yet. We need to do editing source (via text editor) and then compile it again. Another way to check if the compilation was successful by entering the folder where the source files are stored, and type the command

```
ls or dir
```

If the output file that we define or file **a.out** has appeared , then compile successfully. Conversely, if the file has not appeared yet, it means the compilation is fail. Execute the output file can be performed by typing the command

```
./<nama file output>
```

*Example :* **./keluaran**

This above command allows us to see the results of the program have been made.





### 2.2.2. Basic Data Types (Integer, Real, Character, String, Boolean)

#### a. Background

In the Beginning of the discussion of programming, we will first use the basic data types including data type integer, real, char, string and boolean.

#### b. Theory

**Data type** is the data type based on the content and it's characteristic. For example, the same analogy with the case of the daily is gallon of water which just made to accommodate certain types of objects with the objects type is liquid, such as water.

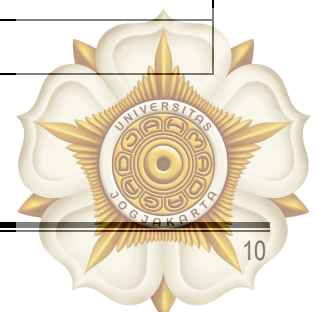
The types of basic data types of programming, such as: Integer, integer type commonly called as an integer, but the integer type does not only consist of an integer, there are other types such as **short** and **long**, which distinguishes these three types is its **range number**.

- **Real**, this type is used to declare value that requires precision numbers with the values behind the comma(fraction value behind decimal point). Such as: double, single, float.
- **Char**, the type of data used to store a character.
- **String**, the type of data in the form of a collection of characters (one or more) that lies between two quotation marks (") in C.
- **Boolean**, data types are used to express the statement is true (true) or incorrect (false).

Reach of each data type are different either of the scope or value of the programming language.

**Table 2.1 Range of data types**

Basic type	Range of points	Digit of Precision
Char	-128 to +127	-
Int	-32768 to +32767	-
Long	-2.147.438.648 to 2.147.438.647	-
Float	3,4E-38 to 3,4E38	6-7
Double	1.7E-308 to 1.7E308	15-16
Long Double	3.4E-4932 to 1.1E4932	19





**Variable** are places to store data of a specified type whose contents can be changed according to the type. Each of variable can only store a single value. So if the value is changed, the previous value is replaced with the new value. Whereas the constant actually is a variable defined by default value at the beginning, and is typically not changed. Variables and constants must be declared in order to the program can allocate a memory to accommodate the specific data and process it in the program to obtain the appropriate output. C++ Language has support variable declarations as well as initialization, for example, is :

```
string  tabelmerk[5]={"adidas  adizero  Indonesia","adidas  climacool  
Vietnam ", "adidas f50 China "};    //array declaration in 5 array,  
    //Each array capable of storing strings  
tabelmerk[3]="Nike mercurial Indonesia"; //initialize each array  
tabelmerk[4]="Nike Air Vietnam";
```

Actually the initialization is also called as an **assignment** that means the process of inserting values into variables with the help of the operator (=). From the above example , the assignment process is entering the data string (character set) into each of **tabelmerk array [0] unto array tabelmerk [4]**.

**Constants** are similar to variables, but has a fixed value. Constants can contain values of interger, float, character, and string. Constant declaration can be done in two ways:

➤ **Using the (#define)**

Constant declarations in this way will be easier to do,because includes a preprocessor directive **# define**. And this syntax is put together with the **# include** statement (**above main ()**). The Syntax format is:

**#define identifier value**

Example of application:

```
#define phi 2.414159265  
#define Newline '\n'  
#define lebar 100
```

Declaration for # define is without featured by = **notation** to enter a value into the identifier and also without ended by a semicolon or **(;)** notation

➤ **Using constants (const)**

With the **const** keyword,the declaration is similar to a variable declaration **plus** coupled with the"const" word and initialized immediately..



Example:

```
const int lebar = 100;  
const char tab = 'p';  
const zip = 912;
```

For the last example, the declaration of variable zip is without data type, then the compiler will automatically enter it into the type of int .

### c. Implementation of Data Type

The most important step in early making a program is to declare a variable that will be used and the type of data type, then do the initialization of these variables, then defines the problem-solving processes that can be kind of calculation formulas or instructions and other commands. Declaring a variable and its data type in C++ language can be seen in the following steps:

#### 1. Task 1 : Make the sum of two integers:

##### 1) Step 1 : Declaration of the variables

*Explanation:* The variables that are used to process the sum of two integers are a and b . Therefore, two numbers are processed are integer, then the type of data used in the variable a and variable b is an **integer**.

##### 2) Step 2 : Initialization

*Explanation:* Initialize the variables a and b is assigning initial values for the two variables that will be used.

##### 3) Step 3 : Process

*Explanation:* Contains the sum process.

##### 4) Step 4 : Finalisasi

*Explanation:* is the step to end the program, including display the results to the output devices (screen), and **return 0** is used to terminate and return a value, because of our program use INT (see the main function), hence this needs the return value. Then we use **return 0** in order to avoid misunderstanding between us with the program..



```

latihan2.cpp
#include <iostream.h>
using namespace std;

int main () {
    //deklarasi variabel :
    int a,b;
    int hasil;
    //inisialisasi :
    a=5;
    b=2;
    //proses :
    hasil=a+b;
    //tampilkan hasil ke layar :
    cout << hasil;
    getch();
    //memberhentikan program
    return 0;
}

```

Figure 2.6 Display of program code

## 2. Task 2 : Compile & Execute program

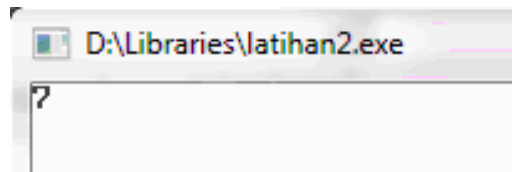


Figure 2.7 Display the results of the program

### 2.2.3. Operator

#### a. Background

To process a more complicated processing sometimes we need a symbol or a sign to treat some variable which is called as operator..

#### b. Theory

Operator is a symbol or a sign that if put between on two operands can yield a result, for example in the mathematical plus sign (+) when placed between two other numbers will produce accretion results rate of two numbers. Plus(+) sign is called an operator.

The operator has several types as follows:



### ○ Arithmetic Operators

Operator	Description	Example
+	Addition(Sum)	a+b
-	Substraction	a-b
*	Multiplication	a*b
/	Division	a/b
%	The rest of division (modulo)	a%b
-	Negation	-a

Negation operator (-) is called an unary operator, as it can requires only a single operand, while the operator% (modulus) is used to search the rest of the division between the two numbers.

Example :  $9 \% 2 = 1$ ,  $9 \% 3 = 0$

### ○ Operator of Relation

Operator	Description	Example	
==	Equal to	a == b	Is <b>a</b> equal to <b>b</b>
!=	Not equal to	a != b	Is <b>a</b> not equal to <b>b</b>
>	Greater than	a>b	Is <b>a</b> greater than <b>b</b>
<	less than	a<b	Is <b>a</b> less than <b>b</b>
>=	Greater than or equal to	a>=b	Is a greater than or equal to b
<=	Less than or equal to	a<=b	Is a less than or equal to b



○ **Operator Increment & Decrement**

Operator	Description	Example	Actual expression
++	Increment	a ++	a=a+1
--	Decrement	b --	b=b-1

○ **Operator Bitwise**

Operator	Description	Example
<<	Shifts n bits to the left ( left shift )	a >> b
>>	Shifts n bits to the right (right shift )	a << b
&	Bitwise AND	a & b
	Bitwise OR	a   b
^	Bitwise XOR	a ^ b
~	Bitwise NOT	~ b

○ **Logical Operator**

Operator	Description	Example
& &	Logical AND	a && b
Logical OR	a    b	
!	Logical NOT	! b

In the above logical operators are often used in selection or looping conditions, for example:

```
If ( (a+b>c) && (a+c>b) && ( b+c>a) )
while (x>=' f' || x<=' c' )
```

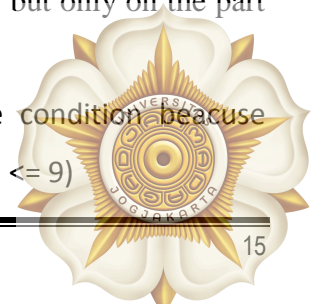
**The assignment operator '='** should not be used in conditions of selection or looping .

Here the example::

```
if (a=b+4) // wrong syntax,
if (a==b+4) // TRUE written syntax.
```

On statement '**for**' it also found the assignment but not on its condition, but only on the part of forward or backward counter. Example::

```
for(int i=0; i<=9; i=i+2) // , i=i+2 not lied in the condition beacuse
separated by sign (;) // But the part that states the condition is on (i <= 9)
```



- **Operator of condition**

Operator of condition is notated by notation '?' that used to obtain the value of two possibilities:

**ungkapan1 ? ungkapan2 : ungkapan3**

When value of ungkapan1 = true, then the value is equal to **ungkapan2**, otherwise is equal to **ungkapan3**

#### **2.2.4. I/O**

##### **a. Background**

Accuracy in the processing of the problem lies in whether the results are in accordance to requirement or not (valid or invalid). To get the expected results, it should be considered the input, process, and output.

##### **a. Theory**

The performed Operation on the input is to read data or value that will be processed. The value of a variable can be early defined in the program or entered by the user from the keyboard by using the existed functions of the library in C++ programming language.

Unlike the operation on the input, the output operation is done to transmit or display data or value to the output device (output device), such as a printer or screen (monitor). Examples of output operation is displaying the sentence to the screen, that usually is done to show commands to enter an input into the program, or display the value of a variable to the monitor by using the existed libraries of C++ programming language, also usually done to show the results of a calculation or the result of a solution . In C++ iostream library, standard input and output operations for programming are supported by 2 data streams: cin for input and cout for output.

*Standard output (cout)*, the use of **cout** stream is coupled with overloaded operator << (**A pair of the "le"s than"**).”Example :

```
cout << "Ka"imat keluaran"; // print the output sentence on the screen.
```

```
cout << 120; // prints number 120 on screen.
```

```
cout << x; // print the contents of the variable x on the screen.
```

Operator << is also Known as the insertion operator, which is used to yield the data that follows it. If a string, then it must be enclosed in double quotes ("), “so is unlike with the variable.



Example :

```
cout << "He"lo"; // prints the output phrase Hello on the screen..  
cout << Hello; // prints the content of Hello variable on the screen..
```

Operator insertion (<<) may be used more than 1 time in the same sentence, for example:

```
cout << "He"lo, " <" "Sa"a " <" " k"limat C++";
```

The above example will display Hello, I am a C ++ sentence on the monitor screen. Benefit from the repetition use of insertion operator (<<) is to display a combination of one or more variables and constants,

Example 1 :

```
cout << "He"lo, umurku" <" umur << "ta"un dan  
aku angkatan" <" angkatan;
```

For instance, **umur** variable is filled with the value **18**, and the **angkatan** variable is charged with **2012**. Then the result of output is:

**Hello, umurku 18 tahun dan aku angkatan 2012**

Example 2 :

```
cout << "Ka"imat Pertama.\n "; cout << "Ka"imat  
Kedua.\nKalimat Ketiga.";
```

Then the resulting output is:

**Kalimat Pertama.**

**Kalimat Kedua.**

**Kalimat Ketiga.**

Beside the new-line character, can also use the **endl** manipulator, for example:

```
cout << "Ka"imat Pertama. " <" endl;  
cout << "Ka"imat Kedua" <" endl;
```

Output :

**Kalimat Pertama.**

**Kalimat Kedua.**

*Standard input (cin)*, its use by adding overloaded extraction operator(>>) on the cin stream that followed by variable that will store the data.



Example:

```
int umur;  
cin >> umur;
```

The above example declares '**umur**' variable in type of int and wait for input from **cin** (keyborad) to be stored in the **umur** variable . **Cin** command will process the input from the keyboard once and the ENTER key must be pressed.The **cin** can also be used for more than one input:

```
cin >> a >> b;
```

*Equivalent to :*

```
cin >> a;  
cin >> b;
```

In this case the data in the input should be 2, one for variables a and the other for b which are written separated by: space, tabular or newline.

Usually **cout** (standard of stream output) is intended to the monitor and **cin** (the standard of stream input) is intended for the keyboard. By using these two streams, then we can interact with the user by displaying messages on the monitor and receive input from the keyboard.

## **b. Implementation of I/O**

Example for implementations of I/O (Input Output), which is a program used to receive input from the students namely nim, name, age, address, and NEM values and displays each of student data themselves to the screen.

### **1. Task 1 : Create a student data program.**

#### **1) Step 1 : Declare the variables used and their data types.**

*Explanation:* The variables that are used to display process of the students data themselves namely the variable **nim**, **name**, and **address** that each of them are variable of **char** data type, these difference only in size. Variable **umur** of integer type, and nem variable of **float** type because it consists of a value that requires number precision after the decimal point.





## 2) Step 2 : Initialization.

*Explanation:* Because the program receives direct input from the user, then it may not be given early initialization. Giving early initialization is adapted to the needs of the processed case.

## 3) Step 3 : The process of solving the problem.

*Explanation:* Reading input from the user by using **cin** function, and displays the results to the screen using **cout** function.

## 4) Step 4 : Finalization.

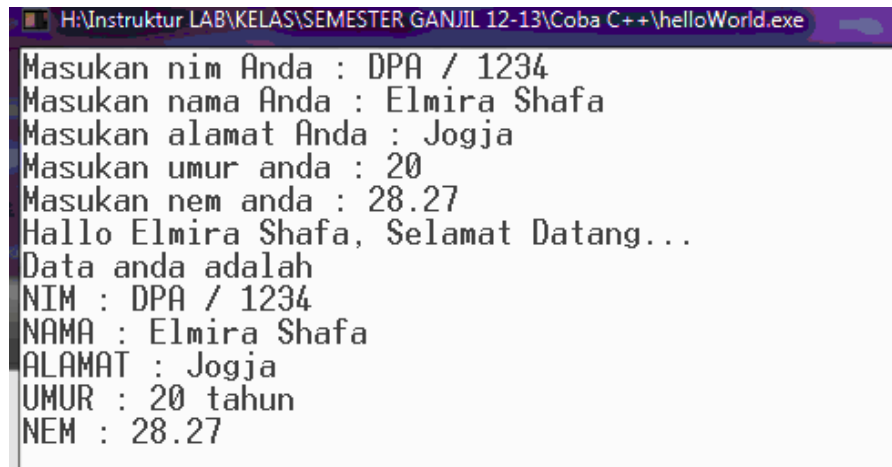
*Explanation:* Stopping the main() function is by using the return 0 command.

```
[*] helloWorld.cpp | Untitled2 |
#include <iostream.h>
#include <conio.h>
using namespace std;
int main() {
    char nim[20]; char nama[20]; char alamat[30]; int umur; float nem; //deklarasi variabel
    cout << "Masukan nim Anda : "; //tampilan ke layar
    //cin.getline(nim,20);
    gets(nim); //menerima masukan variabel nim
    cout << "Masukan nama Anda : ";
    gets(nama);
    cout << "Masukan alamat Anda : ";
    gets(alamat);
    cout << "Masukan umur anda : ";
    cin >> umur; //menerima masukan variabel umur
    cout << "Masukan nem anda : ";
    cin >> nem;
    //menampilkan hasil ke layar
    cout << "Hallo " << nama << ", Selamat Datang..." << endl;
    cout << "Data anda adalah " << endl;
    cout << "NIM : " << nim << endl;
    cout << "NAMA : " << nama << endl;
    cout << "ALAMAT : " << alamat << endl;
    cout << "UMUR : " << umur << " tahun" << endl;
    cout << "NEM : " << nem ;
    getch();
    return 0;
}
```

Figure 2.8 finalization Sourcecode



## 2. Task 2: Compile and Execution



```
H:\Instruktur LAB\KELAS\SEMESTER GANJIL 12-13\Coba C++\helloWorld.exe
Masukan nim Anda : DPA / 1234
Masukan nama Anda : Elmira Shafa
Masukan alamat Anda : Jogja
Masukan umur anda : 20
Masukan nem anda : 28.27
Hallo Elmira Shafa, Selamat Datang...
Data anda adalah
NIM : DPA / 1234
NAMA : Elmira Shafa
ALAMAT : Jogja
UMUR : 20 tahun
NEM : 28.27
```

Figure 2.9 running programs display

### 2.3. Activity

- Student do the task in the implementation of each sub chapter
- Student do the exercise

### 2.4. Exercise

- Make summation program for two value by using input numbers from the user when program run.
- Create a program to calculate the time taken based on the given speed and distance by using the input by the user.
- Create a program to consider a number is odd or even.
- Create a program to determine a number is prime or not.
- Create a program to calculate the area of a circle.

**Make conversion program for currency rupiah to dollars using the input of rupiah nominal by the user, if known 1 USD = Rp 9600, -.**



## CHAPTER III SEQUENCE AND CONTROL STATEMENT

### 3.1. Learning Objectives

1. Explain the logical steps in an algorithm
2. Explain the various control statements in C++
3. Explain how to use the control statement in C++
4. Explain the logic operations to set the conditions in control statements

### Competence

1. Students can resolve the issue in sequence and logically
2. Students understand how to use the control statement
3. Students understand how to set the conditions in the control statement

### 3.2. Theory

#### 3.2.1. Introduction

Algorithm is a sequence of one or more instructions or statements, and each statement is done sequentially in the order of writing. Which is:

1. Each instruction is done one by one
2. Each instruction executed exactly once (no instruction is repeated)
3. Each instruction executed in the same order as it written in the algorithm
4. The end of the last instruction is the end of the algorithm.

For example, suppose we have two cup, say cup A and cup B. Cup A contains coffee and cup B is milk. If we want to switch the content of the two cup, milk on A and coffee on B. How are we going to do that? Consider the following algorithm:

1. Move the content of cup A to B
2. Move the content of cup B to A

What is going on here? Is that solve the problem?

No, we do not solve the problem. We do not exchange the contents of the two cup but mixing both contents. In the end, cup A contains milky coffe while cup B is empty.

Then, what is the right algorithm? Here we need additional cup, say cup C, to temporary hold the content of cup to avoid mixing the contents. Thus, the algorithm will be:

1. Move the content of cup A to C
2. Move the content of cup B to A
3. Move the content of cup C to B

What about now? First we move coffee from cup A to cup C, which is empty, thus not mixing it. Then move milk from B to A, which is also empty after the first step. And finally,



move coffee to cup B. In the end, milk in cup A, coffee in cup B, as expected, and cup C is empty. That solve the problem.

So to switch two cups content, we need an additional cup. Is that all? Consider the following algorithm:

1. Move the content of cup A to C
2. Move the content of cup C to B
3. Move the content of cup B to A

In above algorithm, in the end, cup A contiant milky coffee, cup B and C is empty. We use additional cup as the second algorithm. Then why the result was the same as the first algorithm? We see that the sequence of the second and the third algorithm was different. On the second algorithm we move content B to A, before we moving content cup C to B. While on the third, we move content B to A after moving content C to B.

So, to solve a problem not only we need right technique, but also the right sequence.

A similar problem on a programming algorithm is when we want to swap the contents of two variables, as this code:

```
// Kita ingin menukar isi dari variabel berikut
int x = 10;
int y = 33;
```

So what we need is an auxiliary variable to temporarily store the data of the variable

```
// perlu 1 variabel pembantu
int z = 0;
```

If we print the contents of all the variables in the initial stages, it will be found the following results

```
|inisialisasi
x = 10
y = 33
z = 0
```

The first step in doing this is to move the value of the variable x to variable z.

```
// pindahkan nilai x ke var z
z = x;
```

The contents from each variable after this process is as follows

```
pertukaran pertama x ke z
x = 10
y = 33
z = 10
```



The next step is to move the value of the variable y to variable x

```
// pindahkan nilai var y ke var x
x = y;
```

So that each variable would be

```
pertukaran kedua y ke x
x = 33
y = 33
z = 10
```

The final step is to move the value of the variable z to the variable y

```
// pindahkan nilai var z ke var y
y = z;
```

so the result of this step is the solution of the initial problem, that is

```
pertukaran terakhir z ke y
x = 33
y = 10
z = 10
```

Here's the entire code, and the output of the above issues

```
#include <iostream>

using namespace std;

int main()
{
    // Kita ingin menukar isi dari variabel berikut
    int x = 10;
    int y = 33;

    // perlu 1 variabel pembantu
    int z = 0;

    cout << "inisialisasi" << endl;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "z = " << z << endl;

    // pindahkan nilai x ke var z
    z = x;
```



```

cout << "\npertukaran pertama x ke z" << endl;
cout << "x = " << x << endl;
cout << "y = " << y << endl;
cout << "z = " << z << endl;

// pindahkan nilai var y ke var x
x = y;

cout << "\npertukaran kedua y ke x" << endl;
cout << "x = " << x << endl;
cout << "y = " << y << endl;
cout << "z = " << z << endl;

// pindahkan nilai var z ke var y
y = z;

cout << "\npertukaran kedua z ke y" << endl;
cout << "x = " << x << endl;
cout << "y = " << y << endl;
cout << "z = " << z << endl;

return 0;
}

```

Figure 3.1 Swapping Variables Value Source Code

```

inisialisasi
x = 10
y = 33
z = 0

pertukaran pertama x ke z
x = 10
y = 33
z = 10

pertukaran kedua y ke x
x = 33
y = 33
z = 10

pertukaran kedua z ke y
x = 33
y = 10
z = 10

```

Figure 3.1 Swapping Variables Value Programs When Running



### 3.2.2. Control statement

Control statements can make a program more flexible. With control statements we can define the program line to be executed if a condition is met, and is not executed if the condition is not met. There are two control statements in C++. First, we can use if...else statement and the second is to use switch case.

#### If ... Else

The general form of the if statement is

```
if (kondisi)
{
    statement1;
    statement2;
    ....
    statementN;
}
```

atau :

```
if (kondisi)
{
    statement1;
    statement2;
    ....
    statementN;
}
```

```
else if (kondisi)
{
    statement1;
    statement2;
    ....
    statementN;
}
```

.....

```
else
{
    statement1;
    statement2;
    ....
    statementN;
}
```

The level of if...else statement can be defined as needed.

The condition that must be met can be formed from logical operator and relational operator.



Logical operators that can be used, among other

<i>Symbol</i>	<i>Meaning</i>
&&	AND
	OR
!	NOT

While the relational operator, among other

<i>Symbol</i>	<i>Meaning</i>
==	equal to
>=	greater or equal to
>	more than
<=	less than or equal to
<	less than
!=	not equal to

Sample Program

```
#include <iostream>

using namespace std;

int main()
{
    int b;
    float jumlah = 0;
    cout << "Masukkan nilai b = ";
    cin >> b;

    if (b > 10)
        jumlah = jumlah + b;
    else
        jumlah = jumlah - |b;

    cout << "Jumlah = " << jumlah << endl;

    return 0;
}
```

Figure 3.2 Sample Source Code using relational operators





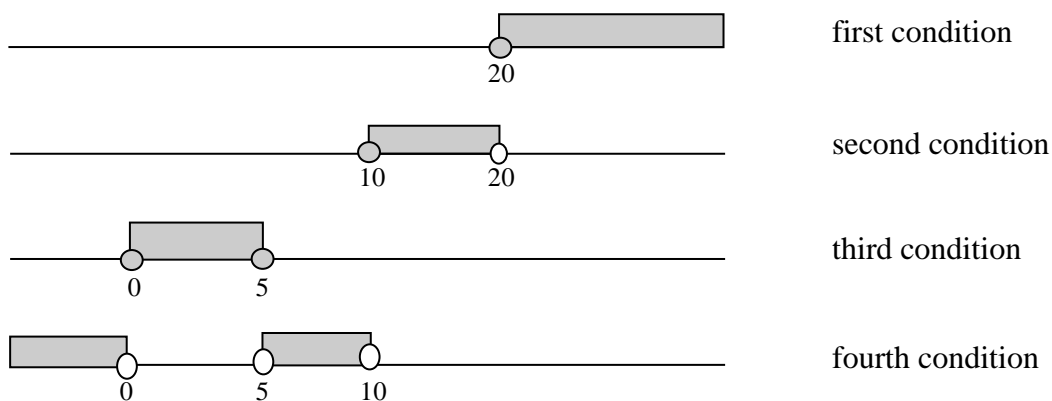
In the above example if the variables are weighted more than 10 then the statement `jumlah = jumlah + b` will be executed. However, if the entered value equal to 10 or less than 10, then the else statement is executed. Because only one line statement after if then braces can be removed.

The compiler will check if condition. If true, then it will execute `jumlah = jumlah + b`, and skip the else statement. However, if false, then the else statement will be checked.

In designing control statement we should use the number line to make it easier to find true area of each condition. For example, suppose we have the following control statement:

```
if (b >= 20)
    ....
else if (b < 20) AND (b >= 10)
    ....
else if (b <= 5) AND (b >= 0)
    ....
else
    ....
```

In above example, there are 4 condition with each number line as follow



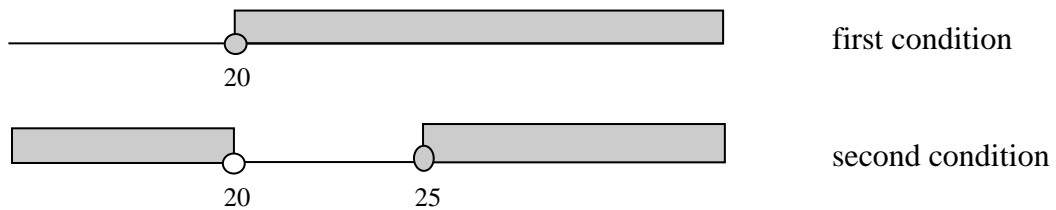
The first condition will be checked first before the others. If the condition is true, then the other condition won't be checked by compiler. However if it false, then the second condition is checked. If it true, then the following condition will be skipped. But if it false, the third condition will be checked. And so on until the fourth condition. This occurs because the control statement are still in one block.

The following example illustrates the importance of using number line

```
if (b >= 20)
    ....
else if (b < 20) AND (b >= 25)
    ....
```



The number line is as follows



There is a regional disparities between the first conditions and the second one. Inequality occurs at  $b \geq 25$ . If both first and second conditions are meets. Only the statement in the first condition that will be executed, because the first conditions encountered first by the compiler and the second condition was skipped.

Block condition begins with the word if, and ends with the words else or new block condition. for example

```
if (kondisi)
    ....

if (kondisi)
    ....
else if (kondisi)
    ....
else if (kondisi) AND (kondisi)
    ....
else
    ....

if (kondisi)
    ....
else if (kondisi)
    ....
```

In the example above there are three blocks of control statement. Block 1 is the first row, block 2 is the line 2-5, and block 3 is the line 6-7. Block condition different with nested control statement. Each block has a equal position.

Here's an example of a nested control statement

```
if (kondisi)
{
    statement;
    ....

    if (kondisi)
        statement;
    else if (kondisi)
        statement;
    else if (kondisi) AND (kondisi)
    {
```



```

        if (kondisi)
            statement;
        else if (kondisi)
            statement;
        else
            statement;
    }
    else
        statement;
}

```

The example above has a different block control statement. But the three blocks of control statement is not the same level. The third block is part of the second block and the second block is part of the first block.

#### Another Sample Program

```

#include <iostream>

using namespace std;

int main()
{
    int input;
    cout << "Masukkan hari ke- = ";
    cin >> input;

    if (input == 1)
    {
        cout << "Anda memasukkan hari minggu";
    }
    else if (input == 2)
    {
        cout << "Anda memasukkan hari senin";
    }
    else if (input == 3)
    {
        cout << "Anda memasukkan hari selasa";
    }
    else if (input == 4)
    {
        cout << "Anda memasukkan hari rabu";
    }
    else if (input == 5)
    {
        cout << "Anda memasukkan hari kamis";
    }
}

```



```

else if (input == 6)
{
    cout << "Anda memasukkan hari jumat";
}
else if (input == 7)
{
    cout << "Anda memasukkan hari sabtu";
}
else
{
    cout << "Bukan input hari/salah masukan";
}
return 0;
}

```

Figure 3.3 Source code example using Conditional IF

This example will be compared with the switch ... case control statement.

### 3.2.3. Switch ... Case

Switch ... case control statement can only be used in condition using equal relation. In the last if ... else example, all the condition use equal relation. Therefore, we can use switch ... case instead of if ... else. Examples are as follows.

```

#include <iostream>

using namespace std;

int main()
{
    int input;
    cout << "masukkan hari ke- ";
    cin >> input;

    switch(input)
    {
        case 1 : cout << "minggu";
        break;
        case 2 : cout << "senin";
        break;
        case 3 : cout << "selasa";
        break;
        case 4 : cout << "rabu";
        break;
        case 5 : cout << "kamis";
        break;
        case 6 : cout << "jumat";
        break;
        case 7 : cout << "sabtu";
        break;
        default : cout << "bukan input hari/salah masukan";
        break;
    }
    return 0;
}

```

Figure 3.3 Source code example using Conditional CASE



switch ... case simplifies repetitive writing if else and conditions. Typographical errors can be reduced. In addition checking the conditions would become easier. There are additional commands ie break. It used to get out of the switch statement if one of the conditions have been met. With the break, when one case is met then the other cases below are not checked. Therefore, with the break we could distinguish whether the condition is still in a block or not.

Default is the replacement of the previous examples else. If no conditions are met then the default will be executed.

In a different example

```
#include <iostream>

using namespace std;

int main()
{
    int input;
    cout << "masukkan hari ke- ";
    cin >> input;

    switch(input)
    {
        case 1 : cout << "minggu";
        case 2 : cout << "senin";
        case 3 : cout << "selasa";
        case 4 : cout << "rabu";
        case 5 : cout << "kamis";
        case 6 : cout << "jumat";
        case 7 : cout << "sabtu";
        default : cout << "bukan input hari/salah masukan";
    }

    return 0;
}
```

Figure 3.4 Source code example using Conditional CASE DEFAULT

Since the break in each statement is omitted, then the blocks of each condition will be expanded include blocks of the underlying condition. If any of these conditions are met then the statement on the underlying condition will be executed as well. For example we inputting 4, then the output will be rabu kamis jumat sabtu bukan input hari/salah masukan, all five from condition 4.

### 3.3. Activity

1. Student try the source code in the figure
2. student do the exercise



### 3.4. Exercise

1. Create a program that can convert a number value that the user inputted into the value of the letter, with conditions:

- The value of 85-100, got A.
- The value of 70-84 got B.
- The value of 40-69 got C.
- The value of 20-39 got D.
- other than that got E.

2. Create a program to determine the greatest number of 3 pieces of numbers inputted by the user.

for example:   enter the 1st number: 15  
                  enter the second number: 7  
                  enter the third number: 22  
                  the greatest number is: 22



## CHAPTER IV RECURRENCE (Looping)

### 4.1. Learning Objectives

1. Describe the a variety of loops are supported by C++
2. Describe how to use a loop in the program
3. Describe the differences of each type of recurrence

### Competence

1. Student is able to use looping in creating program
2. Student know the differences of each type of loop and be able to select the loop that will be used in accordance with their needs

### 4.2. Theory

**Looping**, is section that assigned to conduct activities to repeat a process in accordance with the desired.

The types of looping in C++, among others

- **for** used in a known amount iteration count. (how many times the loop will be done).
- **while**, is function to repeat a statement during the condition is true.
- **do-while**, almost the same as while loops, but the condition of the do-while loop will be checked after the statement is executed. If the condition is true then the statement is executed again, but if the condition is false, the statement is not executed.

### Looping Condition

Conditions used in the iteration has some rules so that there are no error in program. First, the loop condition variable must be of type integer or char. Variable of type float is not recommended though it could. If forced to use a condition variable of type float / fractions, one should check the logic of the conditions carefully.

Especially for loop, the parameters can be removed one, two, or even all of them. But stopping factors should remain there. These factors can be placed in a looping statement.

### Nested Looping

**Nested looping**, is a loop inside another loop. The deeper loop will be processed first until it finish, then the outer loop will be increased, then work again on a deeper iteration starting from the initial value onwards. Looping inside another loop is allowed in almost all programming languages.



Nested looping example can be seen in the following programs:

```
#include <iostream.h>
int main()
{
    int nested1, nested2;
    int awal1 = 1, awal2;
    cout << "masukkan nilai nested1 = ";
    cin >> nested1;
    cout << "\nmasukkan nilai nested2 = ";
    cin >> nested2;

    if ((nested1 > 0) && (nested2 > 0))
    {
        while (awal1 <= nested1)
        {
            for(awal2=1; awal2 <= nested2; awal2++)
            {
                cout << "Perulangan ke-" << awal1 * awal2 << endl;
            }
            awal1++;
        }
    }
}
```

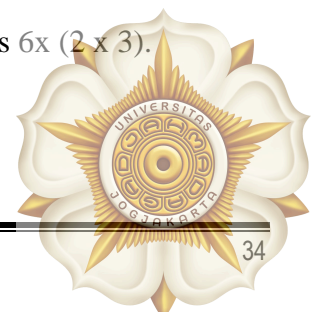
**Figure 4.1 Source code example using While and For Nested looping**

In the example above, there is a loop inside another loop. All forms of looping can be put on another looping form. As in the example, for loop was inside the while loop. Statement line 18, will run as many as nested1 x nested2 (according to user input).

Course of the example is as follows. First, the value for nested1 and nested2 variable inserted by the user are 2 and 3. Then if condition will check whether the value of each nested is in positive integer form. If true then the while loop will be executed. Inside the while loop condition will be checked whether the value awal1 less than the value nested1 (2). Because the condition is true, statement inside the while loop is executed. Inside the while loop statement, there are loop again, then the looping condition is checked. Is awal2 value less than nested2? Because the condition is true then for statement is executed. Which displays a message to the screen Perulangan ke-...

For statement will be executed until the value of awal3 more than nested2 value. In other words the for loop statement, run 3 times. Meanwhile the while loop statement is executed as much as 2x. In conclusion total statement line 28 executed as much as 6x (2 x 3).

Program output as follows:





```

masukkan nilai nested1 = 2
masukkan nilai nested2 = 3
Perulangan ke-1
Perulangan ke-2
Perulangan ke-3
Perulangan ke-2
Perulangan ke-4
Perulangan ke-6

```

**Figure 4.2 Display of Running Program using While and For Nested looping**

### 4.3. Activities

#### 1). Using the FOR loop :

##### 1. Task 1 : Creating an algorithm

##### 1) Step 1 : Variable Declaration

Explanation: Declaring a variable to be used as an index to display the row number 1, for example, i, of type integer.

```
i : integer
```

##### 2) Step 2 : Initialitation

Explanation: Filling the variable i to the initial value, aims to create a pre-looping.

```
i ← 1
```

##### 3) Step 3 : Process

Explanation: Contains all the parts of the loop that is performed repeatedly. This process is displays number 1 that repeated 6 times. In this step, iteration occurs in the loop, which is an increament condition so that the loop can continue to run.

```

for i ← 1 to 6 do
    write ("1")
end for

```

##### 4) Step 4 : Finalization

Explanation: Stopping condition of the loop is very important to prevent the loop that runs continuously.



## 2. Task 2 : Create the program

### 1) Step 1 : Variable declaration

```
int i;
```

### 2) Step 2 : Initialization + Process + Finalization

```
for(i=1; i<=6; i++) {  
    cout<<"1";  
}
```

```
#include <iostream.h>  
#include <conio.h>  
using namespace std;  
int main () {  
    int i;  
    for (i=1; i<=6; i++)  
        cout<<"1";  
    getch();  
    return 0;  
}
```

Figure 4.3 Source code example using FOR looping

## 2. Task 3 : Compilation and program execution

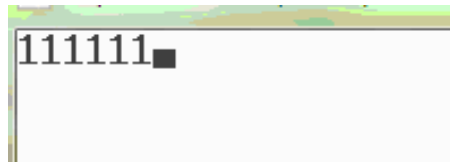


Figure 4.4 Display of Running Program using FOR looping

## 2). By using WHILE loop :

### 1. Task 1 : Creating an Algorithm

#### 1. Step 1 : Variable declaration

Explanation: Declaring a variable to be used as an index to display the row number 1, for example, i of type integer.

```
i : integer
```

#### 2. Step 2 : Initialitation

Explanation: Filling the variable i to the initial value, aims to create a pre-looping conditions.

```
i ← 1 ;
```



### 3. Step 3 : Process

Explanation: Contains all the parts of the loop that is performed repeatedly. This process is displays number 1 that repeated 6 times. In this step, iteration occurs in the loop, which is an increament condition so that the loop can continue to run.

```
while (i<=6) do {  
    cout<<"1";  
}
```

### 4. Step 4 : Finalization

Explanation: Stopping condition of the loop is very important to prevent the loop that runs continuously.

```
i ← i+1; → iterasi  
end while;
```

## 2. Task 2 : Creat the program

### 1. Step 1 : Variable declaration

```
int i=1;
```

### 2. Step 2 : Initialization + Process + Finalization

```
while ((i<=6) {  
    cout<<"1";  
    i=i+1;  
}
```

```
#include <iostream.h>  
#include <conio.h>  
using namespace std;  
int main() {  
    int i;  
    i=1;  
    do {  
        cout<<"1";  
        i=i+1;  
    }  
    while (i<=6);  
    getch();  
    return 0;  
}
```

Figure 4.5 Source code example using While looping



### 3. Task 3 : Compilation and program execution

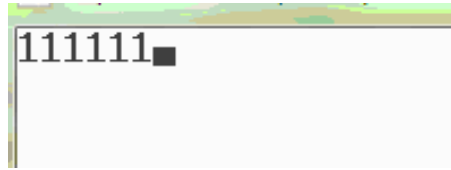


Figure 4.6 Display of Running Program using While looping

#### 3). By using DO-WHILE loop :

##### 1. Task 1 : Create an algorithm

###### 1. Step 1 : Variable declaration

Explanation: Declaring a variable to be used as an index to display the row number 1, for example, i of type integer.

```
i : integer;
```

###### 2. Step 2 : Initialization

Explanation: Filling the variable i to the initial value, aims to create a pre-looping conditions.

3.  $i \leftarrow 1$

###### 4. Step 3 : Process

Explanation: Contains all the parts of the loop that is performed repeatedly. This process is displays number 1 that repeated 6 times. In this step, iteration occurs in the loop, which is an increament condition so that the loop can continue to run.

```
repeat {  
    write("1");  
    i ← i+1;  
until (i=6)
```

###### 5. Step 4 : Finalization

Explanation: Stopping condition of the loop is very important to prevent the loop that runs continuously.

##### 1. Task 2 : Create the program

###### 1. Step 1 : Variable declaration

```
int i=1;
```



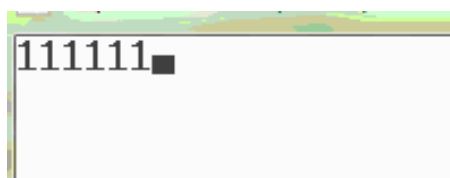
## 2. Step 2 : Initialization + Process + Finalization

```
do {  
    cout<<"1";  
    i=i+1;  
}  
while (i<=6);  
return 0;  
}
```

```
#include <iostream.h>  
#include <conio.h>  
using namespace std;  
int main() {  
    int i;  
    i=1;  
    do {  
        cout<<"1";  
        i=i+1;  
    }  
    while (i<=6);  
    getch();  
    return 0;  
}
```

Figure 4.7 Source code example using While looping

## 3. Task 3 : Compilation and program execution



111111

Figure 4.8 Display of Running Program using While looping



#### 4.4. Exercise

1. Create a program to display a sequence of numbers using for loop, while, do. . while

1 2 3 4 5 6 7 8 9

2. Create a program using for loop, while loop and do-while loop to display rows of letters

A B C D E F . .

3. Create a program to display stars, within the limits of the amount of numbers that inputed by user, for example, the input limits is 4, as in the following figure

```
*           * * * *           *
**          * * *           **
***         * *           ***
****        *           ****
                    ***
                    **
                    *
```

4. Create a program to display multiplication table, as an example:

```
* 1 2 3 4 5
1 1 2 3 4 5
2 2 4 6 8 10
3 3 6 9 12 15
4 4 8 12 16 20
5 5 10 15 20 25
```



## CHAPTER V

### ARRAY DATA TYPE

#### 5.1. Learning Objectives

1. Describe the nature of the array
2. Explain the intricacies of the array
3. Explains how to use the array

#### Competence

1. Student understand the meaning and intricacies of the array
2. Student know how to use arrays in the program

#### 5.2. Teory

##### 1. Array

If we discuss about the array, to be familiar with the basic concepts, we can analogize with the following example:

Name	Anik	Winy	Sita	Sulvi	Anin	Vemi
Grade	80	90	95	95	80	85

**Figure 5.1 Illustration image of array in a data value**

So it can be concluded that array is a collection of many variables with the same data type, indicated by an index on each element. Functions and main features of the array are storing a series of elements of the same type and has indexes which can be accessed directly or randomly. There are 2 types of array , one dimensional arrays and arrays of two/multi-dimensional so-called matrix. The difference, multi-dimensional arrays index has more than one type or the components type are another array form.

1. **Step 1 : Array Declaration and Initialization**, Forms a variable of type array first, including the name, size, and content of data type.

```
Type name [element];
```

or

```
Type name [element] =  
{element1,element2,element_n};
```



for example :

	1	2	3	4	5	6
Nilai	80	90	95	97	88	85

Sequence of value above can be declared using the array type :

```
int nilai [6];
```

or

```
int nilai [6] = {80,90,95,80,85};
```

	nilai[0]	nilai[1]	nilai[2]	nilai[3]	nilai[4]	nilai[5]
Nilai	80	90	95	97	88	85

2. **Step 2 : Accessing Array**, Perform operations on each element of the data set individually. Format array access is defined as follows.

```
name [index];
```

For example, to store the value of the second row, 90, of these data, it can be written in the following form

```
nilai[1] = 90;
```

The order of elements in the array always starts from 0, then the value of 90 referred by the first index, while the value of 97 displayed by the third index

```
read nilai[3];
```

## 2. Multidimensional arrays

**Matrix**, is an array that have 2 or more rows and also 2 or more columns, or more precisely, is an array that is inside another array, depending on how its use. For example, there is a two-dimensional array with size 2x2 is shown in the following example:

1,1	1,2
2,1	2,2

**Figure 5.2 Illustration of the matrix**





For example :

	0	1	2	3
0				
1				
2				

**1) Step 1 : Array Declaration and Initialization,**

```
Type name [rows_element][column_element];  
Int nilai[3][4];
```

**2) Step 2 : Accessing Array,**

```
name [row_index][column_index];
```

For example, to store values in the second row and the third column, it can be written as follows

	0	1	2	3
0				
1			80	
2		90		

```
nilai[1][2] = 80;
```

To display the values 90 in the third row and the second column can be written as follows :

```
read nilai[2][1];
```

**3. Array of Char**

In C++ string data type can be obtained by defining an array of char. Because it is essentially strings is a collection of variable of type char. Although the form of an array, the treatment of an array of type char a little more special. Since there are several functions that can be used to directly manipulate this array. While on an array of non char there is no function to manipulate. Functions for an array of type char is placed in file include string.h.



How to declare a string? Just like with a regular array but the data type is char.

Here's an example

```
#include <iostream.h>
int main()
{
    char kata1[5];
    char kata2[] = {'B', 'E', 'L', 'A', 'J', 'A', 'R'};

    for (int i=0; i<5; i++)
    {
        kata1[i] = 'A'+i;
        cout << kata1[i];
    }
    cout << endl;
    for (int i=0; i<7; i++)
    {
        cout << kata2[i];
    }
    cout << endl;
    system ("pause");
    return 0;
}
```

Figure 5.3 Source code example using Array Data Type

The above example is similar to the examples in the previous chapter. Char data type is a kind to an integer data type. Because the char data type is filled by the ASCII characters which encoded with an integer number from 0-127. In the first iteration shown that kata1 element is filled with character A (ASCII = 65), B (ASCII = 66), C (ASCII = 66), D (ASCII = 66), and E (ASCII = 66) successively. The output of the above program is:



```
ABCDE
BELAJAR
```

Figure 5.4 Display of Running Program using Array Data Type

### Functions in string.h

Just as with the other include files, string.h is located in / usr / include. We will see what functions are supported by string.h..



```

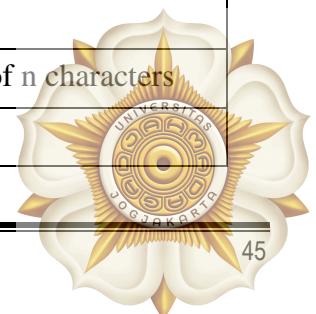
/* Copy SRC to DEST. */
extern char *strcpy (char *__restrict __dest, __const char
*__restrict __src)
    __THROW;
/* Copy no more than N characters of SRC to DEST. */
extern char *strncpy (char *__restrict __dest,
    __const char *__restrict __src, size_t __n)
    __THROW;
/* Append SRC onto DEST. */
extern char *strcat (char *__restrict __dest, __const char
*__restrict __src)
    __THROW;
/* Append no more than N characters from SRC onto DEST. */
extern char *strncat (char *__restrict __dest, __const char
*__restrict __src,
    size_t __n) __THROW;
/* Compare S1 and S2. */
extern int strcmp (__const char *__s1, __const char *__s2)
    __THROW __attribute_pure__;
/* Compare N characters of S1 and S2. */
extern int strncmp (__const char *__s1, __const char *__s2,
size_t __n)
    __THROW __attribute_pure__;
/* Return the length of S. */
extern size_t strlen (__const char *__s) __THROW
    __attribute_pure__;
__END_NAMESPACE_STD

```

These seven functions is the most widely used for manipulating strings. If you are using Microsoft Visual C++, there are more strings manipulation functions, such as strstr,strupr, and strlwr.

Table 5.1 Default Function to processing words

<i>function</i>	<i>Parameter</i>	<i>Description</i>
strcpy	source, dest	copying words from the source to the dest
strncpy	source, dest, n	copying word from source to dest of n characters
strcat	source, dest	adding strings source to strings dest at the end of the dest
strncat	source, dest, n	add strings source to dest at the end position as much as n character
strcmp	s1, s2	comparing string s1 with string s2
strncmp	s1, s2, n	comparing strings of s1 with the s2 strings of n characters
strlen	S	showing the s strings length



The above functions can be used in string type. In the previous example, we display the word cout BELAJAR by doing one at a time from the array elements. To be practical, we will change the previous example with the following program :

```
#include <iostream.h>

int main()
{
    char kata1[6];
    char kata2[] = "BELAJAR";

    strcpy(kata1, "ABCDE");
    cout << kata1 << endl;
    cout << kata2 << endl;

    return 0;
}
```

**Figure 5.5 Source code example using Array Data Type**

Seen that even though strings is an array but it has a different treatment. Declaration of an array of char as well as filling it, different from the declaration as well as filling of non char array value. the program Output is as follows :



```
ABCDE
BELAJAR
```

**Figure 5.6 Display of Running Program using Array Data Type**

### **Capturing Character Input**

Some time we may require our program to capture input from the user in the form of characters. To capture the characters, we can use the same command as capturing input of integer or float. Here's an example program :



---

```

#include <iostream.h>

int main()
{
    char kata1[6];
    char kata2[] = "BELAJAR";
    char kata3[15];

    cout << "Masukkan suatu kata dengan panjang 6 = ";
    cin >> kata1;
    cout << kata1;

    strncpy(kata3, kata1, 3);
    cout << "\n" << kata3 << endl;
    cout << strncmp(kata3, kata1, 3) << endl;
    cout << "Panjang kata 2 adalah = " << strlen(kata2);
    system ("pause");
    return 0;
}

```

**Figure 5.7 Source code example to capture character input**

The output of the program is :

```

Masukkan suatu kata dengan panjang 6 = BELAJAR
BELAJAR
BEL
0
Panjang kata 2 adalah = 3Press any key to continue . . . ■

```

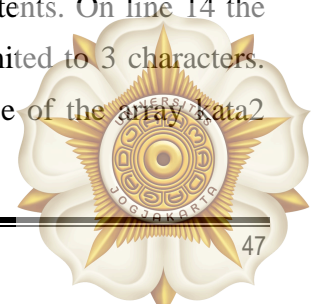
**Figure 5.6 Display of Running Program to capture character input**

In the program there are 3 arrays respectively kata1, kata2, and kata3. Then at the 9th row the program ask for input from the user in the form of words. Characters that can be received only 6 units. Although the user enter as many as 8 characters, only the first 6 characters are stored. How to receive user input in the form of character is with the command

```
cin >> kata1;
```

Although kata1 is an array but accessing char array type can omit the index information. This facility specifically for arrays of char only. On line 10 as well . To display a string <array of char> not need to display one at a time, but can display all at once by omitting array index is concerned.

Row 12 shows how to use *strncpy* function. The contents of the array character copied from array kata1 to larik3 just as much as 3 characters, then display its contents. On line 14 the contents of kata1 and kata3 is compared. But the comparison was limited to 3 characters. The result is a value of 0 which means 'true'. Finally, display the size of the array kata2 which amounted 7 characters.



The conclusion, Ways access an array of char with the non char is different. C++ provides special facilities for an array of char. This is to facilitate programmers of using string data type. Accessing by eliminating the array index. The secret is, every char array that we created is automatically added null character by the C++ which means mark the end of the string..

Like the contents of kata2, by the C++ memory allocation to be :

B E L A J A R /0

the last character is the final mark of the kata2 string. Therefore if done cout then C + + will display the contents of the array by looping kata2 until a null character is found. That why array index when accessing array of char can be removed. Because of its benchmark to the null character.

### 5.3. Activity

- a. Student do all tasks
- b. student do the exercise

### 5.4. Exercise

1. Create a program to sum the numbers that inserted by the user.

```
masukkan jumlah angka yang akan dijumlahkan : 4
masukkan angka ke-0 : 1
masukkan angka ke-1 : 2
masukkan angka ke-2 : 4
masukkan angka ke-3 : 3
10
```

2. Create a program to convert decimal numbers into binary. The conversion results can be directly displayed or stored in an array of char variable



## CHAPTER VI

### STRUCT DATA TYPE

#### 6.1. Learning Objectives

1. Describe the nature of the structure
2. Explain the intricacies of structure
3. Explains how to use the structure

#### Competence

1. Student understand the meaning and intricacies of the architecture
2. Student knows how to use the program's structure

#### 6.2. Teory

If in the previous case, namely the array, each component uses the same data type, it is different in a struct that may has components that have different data types. For example, a record of one's data, which consists of names, addresses, ages and department. All data is collected in a single record with the name, address, age, and department as its fields. So this record is an extension of the array data type.

##### 1 Struct declaration

Form of a prior record, covering the fields that are in the record along with the data type for each field.

```
struct structName {  
    datatype fieldName1;  
    datatype fieldName2;  
    datatype fieldName3;  
};  
e.g :  
struct StudentRec {  
    string name;  
    string idNum;  
    float gpa;  
};
```

Form variable with the record type.

```
structName variableName;  
e.g :  
StudentRec theStudent;
```



## 2 Accessing struct

Perform operations on each element of the individual record. For example, the operation to fill the value of each element. Specific values can be assigned, with the rules to reference field of a record.

```
variable_name.field_name = nilai;  
e.g :  
theStudent.name = "Sally";  
or  
cin >> theStudent.idNum;
```

Menampilkan data yang ada didalam record .

```
cout<<variable_name.field_name;  
e.g :  
cout<< theStudent.gpa;
```

## 3 Nested struct

Elements of a struct can also be another struct. An example can be seen below.

```
struct fullname {  
    string firstname;  
    string lastname;  
};  
  
struct StudentRec {  
    fullname name;  
    string idNum;  
    float gpa;  
};
```

## 4 Array of Struct

Elements of an array can also be a struct. An example can be seen below :

```
struct fullname {  
    string firstname;  
    string lastname;  
};  
  
struct StudentRec {  
    fullname name;  
    string idNum;  
    float gpa;  
};  
  
StudentRec theStudent[10];
```





### 6.3. Activities

#### 1. Task 1 : Make a program to store personal information with struct.

##### 1. Step 1 : Struct Declaration

```
struct fullname {
    string firstname;
    string lastname;
};
struct StudentRec {
    fullname name;
    string idNum;
    float gpa;
};
```

##### 2. Step 2 : Variable declaration and initialization

```
StudentRec theStudent[10];
```

##### 3. Step 3 : Process

```
int main() {
    cout<<"Masukkan banyaknya mahasiswa : ";
    cin>>n;
    cout<<"Data mahasiswa"<<endl;
    for (int i=0;i<n;i++) {
        cout<<"Nama depan : ";
        cin>>theStudent[i].name.firstname;
        cout<<"Nama belakang : ";
        cin>>theStudent[i].name.lastname;
        cout<<"Nim : ";
        cin>>theStudent[i].idNum;
        cout<<"IPK : ";
        cin>>theStudent[i].gpa;
    }
}
```

##### 4. Step 4 : Finalization

```
cout<<"Data mahasiswa"<<endl;
for (int i=0;i<n;i++) {
    cout<<theStudent[i].name.firstname;
    cout<<theStudent[i].name.lastname;
    cout<<theStudent[i].idNum;
    cout<<theStudent[i].gpa;
}
return 0;
}
```



```

#include <iostream.h>
using namespace std;

struct fullname {
    string firstname;
    string lastname;
};

struct StudentRec {
    fullname name;
    string idNum;
    float gpa;
};

StudentRec theStudent[10];

int main() {
    int n;
    cout<<"Masukkan banyaknya mahasiswa : ";
    cin>>n;
    cout<<"Data mahasiswa"<<endl;

    for (int i=0;i<n;i++) {
        cout<<"Nama depan : ";
        cin>>theStudent[i].name.firstname;
        cout<<"Nama belakang : ";
        cin>>theStudent[i].name.lastname;
        cout<<"Nim : ";
        cin>>theStudent[i].idNum;
        cout<<"IPK : ";
        cin>>theStudent[i].gpa;
    }
    cout<<endl;
    cout<<"Data mahasiswa"<<endl;
    for (int i=0;i<n;i++) {
        cout<<theStudent[i].name.firstname<<ends;
        cout<<theStudent[i].name.lastname<<endl;
        cout<<theStudent[i].idNum<<endl;
        cout<<theStudent[i].gpa<<endl;
    }
    system("Pause");
    return 0;
}

```

Figure 6.1 Source code example using struct



## Task 2 : Program compilation and execution

```
Data mahasiswa
Nama depan : Anik
Nama belakang : Budiati
Nim : 12345
IPK : 3.00
Nama depan : Dwiny
Nama belakang : Meidelfi
Nim : 13456
IPK : 3.05
Nama depan : Rosita
Nama belakang : Yanuarti
Nim : 15432
IPK : 3.10

Data mahasiswa
Anik Budiati
12345
3
Dwiny Meidelfi
13456
3.05
Rosita Yanuarti
15432
3.1
Press any key to continue . . .
```

Figure 6.2 Display of Running Program using struct

### 6.4. Exercise

Create a program (by using arrays and structs) which asks the student data (nim, UTS values and UAS value) of N inputs (N is determined by the users themselves, maximum 50), then calculate the average value of each student.

For example :

enter the number of students : 1

enter the nim : 412

enter UTS value : 72

enter UAS value : 74

-----

nim : 412

average : 73



## CHAPTER VII

### SUBPROGRAM AND FUNCTION

#### 7.1. Learning Objectives

- student can differ the function with return value and function without return value
- student can implement recursion in mathematic case

#### 7.2. Theory

In C++, program is a collection of functions that are defined either directly in the program or stored in a header file. C++ itself has main function called *main()*. The *main()* function always presents in every C++ program and the compiler will execute the program through the commands contained in this main function.

Functions are Subprograms and useful to make the program more modular so it can be easily understood and can be used again, either for the program itself or for other programs that have the same process.

A function contains statements that are packed in a name. Furthermore this name can be called several times in several positions in the program. Functions make easier to develop a program and save the size of the program.

##### 7.2.1. Fuction without Return Value

C++ does not have a procedure as in Pascal programming language. In pascal language, procedures are defined as a process that does not return a value. In C ++, we have to create a function with a void type, which means that it does not need a *return* value. The general syntax of the function without return value is as follows

```
void nama_fungsi (parameter1,parameter2,...){  
    // statemen yang akan dilakukan.  
    //....  
    //.....  
}
```

For example, below is a function that can make a line. The syntax of the program is as follows



```

#include <iostream>
using namespace std;
void garis(int n); //prototipe fungsi

int main(){
    garis(50);
    cout<<"Program Percobaan Fungsi"<<endl;
    garis(40);
    cout<<"Fungsi digunakan untuk menghemat program."<<endl;
    cout<<"Fungsi dapat dipanggil berkali-kali dalam program"<<endl;
    garis(30);
    return 0;
}

void garis(int n){
    for(int i=1;i<=n;i++){
        cout<<"-";
    }
    cout<<endl;
}

```

Commonly, the function accepts input called arguments or parameters. Input/parameters processed by the function and generate a *return value*.

Declaration of Function is known as a function prototype.

The prototype is:

- Name of function
- Type of function return value
- The number and type of arguments

And ends with a semicolon (;) as in the variable declaration. Some Benefits of prototype function is to ensure the type of the argument that passed to the function call is really appropriate. Every function that is called in the program must be defined.

If the function has no return value, then the type of the return value is void. Function without a return value does not require a *return* statement in it's definition.

In C++, function arguments can have default values. Here's an example..



```

#include <iostream>
using namespace std;

void tulis(int jml = 2);

int main() {
    tulis(5);
    return 0;
}

void tulis(int jml) {
    for(int i=0; i<jml; i++)
    {
        cout<<"C++"<<endl;
    }
}

```

**Figure 7.1 Source code example using function**

For a better understanding of the functions that have default parameter values, please try the following code and explain its output.

```

#include <iostream>
using namespace std;

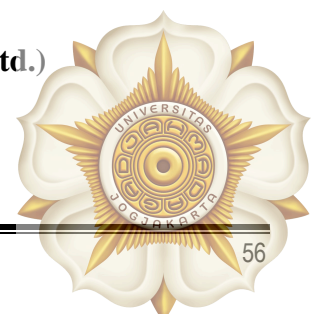
void cetak(char karakter = '-', int jml = 10);

int main() {
    cetak('*', 5);
    cetak('+');
    cetak();
    return 0;
}

void cetak(char karakter, int jml) {
    for(int i=0; i<jml; i++){
        cout<<karakter;
    }
    cout<<endl;
}

```

**Figure 7.1 Source code example using function (contd.)**



### 7.2.2. Function with a return value

In contrast to the void type function, this function is useful to perform a process that can return a value. When making this function we have to define the data type of the value to be returned. Here is the general form of a function that has a return value.

```
tipe_data_kembalian nama_fungsi (params,...){  
    //statemen  
    return nilai_yang_dikembalikan;  
}
```

We call this function the same way as calling the function without return value. For better understanding, study the following example.

```
#include<iostream>  
using namespace std;  
  
double TestFungsi(double x);  
  
int main() {  
    return 0;  
}  
  
double TestFungsi(double x) {  
    return (3.14 * x);  
}
```

### Pass By Reference

In C++, reference is used to provide alias name of the variable. The declaration is as follow:

```
int &ref = nama_variabel;
```

After the declaration, *ref* become a name alias of *nama\_variabel*. Changing the value of the *nama\_variabel* can be done through the *nama\_variabel* itself or through *ref* reference.



```

#include <iostream>
using namespace std;

int main() {
    int i;
    int &r = i; //deklarasi referensi
    i = 10;
    cout<<"i = "<<i<<endl;
    cout<<"r = "<<r<<endl;
    r = 15;
    cout<<"i = "<<i<<endl;
    cout<<"r = "<<r<<endl;
    return 0;
}

```

By using references, an argument value of the function can be changed inside of the function. Before we talk about it, consider the following example:

```

#include <iostream>
using namespace std;
void tukar(int a, int b);

int main(){
    int a = 10;
    int b = 20;
    cout<<"main ()"<<endl;
    cout<<"a = "<<a<<endl;
    cout<<"b = "<<b<<endl<<endl;
    tukar(a, b);
    cout<<"main ()"<<endl;
    cout<<"a = "<<a<<endl;
    cout<<"b = "<<b<<endl<<endl;
    return 0;
}

void tukar(int a, int b){
    int tmp;
    tmp = a;
    a = b;
    b = tmp;
    cout<<"tukar ()"<<endl;
    cout<<"a = "<<a<<endl;
    cout<<"b = "<<b<<endl<<endl;
}

```





In the program, although the value of a and b are exchanged in the tukar() function, but the actual value of a and b in the main() function has not been exchanged. By using references, values of a and b in the main() function can be exchanged inside of tukar() function. Changes need to be made in the function prototype and function definition.

```
#include <iostream>
using namespace std;

void tukar(int &a, int &b);

int main() {
    int a = 10;
    int b = 20;
    cout<<"main () "<<endl;
    cout<<"a = "<<a<<endl;
    cout<<"b = "<<b<<endl<<endl;
    tukar(a, b);
    cout<<"main () "<<endl;
    cout<<"a = "<<a<<endl;
    cout<<"b = "<<b<<endl<<endl;
    return 0;
}

void tukar(int &a, int &b) {
    int tmp;
    tmp = a;
    a = b;
    b = tmp;
    cout<<"main () "<<endl;
    cout<<"a = "<<a<<endl;
    cout<<"b = "<<b<<endl<<endl;
}
```

### 7.2.3. Recursion

Function in C++ can be used in recursion. This means a function can call it's own self. Examples of the application is to calculate the value of powers:  $x^n$  where n is positive integer.

Solutions to that problems is:

if  $n = 1$              $\rightarrow x^n = x$

else                 $\rightarrow x^n = x * x^{n-1}$



```

#include <iostream>
using namespace std;
long int pangkat(int x, int n);
int main() {
    int x, n;
    long int hasil;
    cout<<"Menghitung x^n"<<endl;
    cout<<"Input x : ";
    cin>>x;
    cout<<"Input n : ";
    cin>>n;
    hasil = pangkat(x, n);
    cout<<x<<"^"<<n<<" = "<<hasil;
    return 0;
}
long int pangkat(int x, int n){
    if (n == 1)
        return(x);
    else
        return(x * pangkat(x, n-1));
}

```

### 7.3. Activity

- Student do exercise
- Student try the source code in all figures

### 7.4. Exercise

- Write a program with some function to calculate simple mathematic operation, namely addition or difference!
- Write a program with recurrence function to calculate  $x!$  ( $x$  factorial)!
- Write a program with recurrence function to calculate GCD (Greater Common Division)!



## CHAPTER VIII

### SORTING ALGORITHM & SEARCH ALGORITHM

#### 8.1. Learning Objectives

1. Describe sorting algorithms
2. Describe searching algorithms

#### Competence

1. Students can describe sorting algorithm step by step
2. Students understand Searching algorithm

#### 8.2. Theory

Sorting the data is important in real life for easy data management. Ordering can be done in ascending or descending order. For example, if the data points as follows.

5	3	7	2	0	9	4	1	8	6
---	---	---	---	---	---	---	---	---	---

If the data is sorted in ascending would be as follows.

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

If the data is sorted in descending would be as follows.

9	8	7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---	---	---

There are several sorting methods, such as :

- Insertion Method (*insertion sort*)
- Selection Method (*selection sort*)
- Bubble Method (*bubble sort*)
- Merge Method (*merge sort*)
- Quick Method (*Quick Sort*)

In the discussion below, we will describe sorting algorithms, as well as application in C++.

##### 8.2.1. Insertion Method (*Insertion Sort*)

Direct insertion method is the sorting method that takes an inserted data in the sorted data and **shifts the greater data than the inserted data** so that inserted data can be placed inline in the proper place. For example, if there is an array that contains the following rates of integer.

5	3	7	2	0	9	4	1	8	6
---	---	---	---	---	---	---	---	---	---

If the above data will be sorted in ascending order using the direct insertion method then the process is as follows.



Inserted Data	Sorting Result										
3	<table><tr><td>5</td><td>3</td><td>7</td><td>2</td><td>0</td><td>9</td><td>4</td><td>1</td><td>8</td><td>6</td></tr></table> <p>In the 1st iteration, the index[1] data is used as the inserted data, then compared with previous data. When there is a greater data than the inserted data, so the data(5) must be shifted one place to rightside, and the inserted data moved to the leftmost position</p>	5	3	7	2	0	9	4	1	8	6
5	3	7	2	0	9	4	1	8	6		
7	<table><tr><td>3</td><td>5</td><td>7</td><td>2</td><td>0</td><td>9</td><td>4</td><td>1</td><td>8</td><td>6</td></tr></table> <p>In the 2nd iteration, the data in index[2] is used as the inserted data, then compared with previous data. If the previous data is no greater than the inserted data,so no data to be shifted to the rightside.</p>	3	5	7	2	0	9	4	1	8	6
3	5	7	2	0	9	4	1	8	6		
2	<table><tr><td>3</td><td>5</td><td>7</td><td>2</td><td>0</td><td>9</td><td>4</td><td>1</td><td>8</td><td>6</td></tr></table> <p>In the 3rd iteration, the index[3] data is used as the inserted data, and compared with previous data. There are three greater data than the inserted data, so then the three data must be shifted one place to the rightside and the inserted data positioned to the leftmost place.</p>	3	5	7	2	0	9	4	1	8	6
3	5	7	2	0	9	4	1	8	6		
0	<table><tr><td>2</td><td>3</td><td>5</td><td>7</td><td>0</td><td>9</td><td>4</td><td>1</td><td>8</td><td>6</td></tr></table> <p>In the 4th iteration, the index[4] data used as the inserted data, and compared with previous data. There are four greater data than the inserted data, so the four data must be shifted one place to rightside and then the inserted data positioned to the leftmost place.</p>	2	3	5	7	0	9	4	1	8	6
2	3	5	7	0	9	4	1	8	6		
9	<table><tr><td>0</td><td>2</td><td>3</td><td>5</td><td>7</td><td>9</td><td>4</td><td>1</td><td>8</td><td>6</td></tr></table> <p>In the 5th iteration, the index[5] data is used as the inserted data, and compared with previous data. If in the previous data there are no greater than the inserted data ,so no data to be shifted to the rightside.</p>	0	2	3	5	7	9	4	1	8	6
0	2	3	5	7	9	4	1	8	6		
4	<table><tr><td>0</td><td>2</td><td>3</td><td>5</td><td>7</td><td>9</td><td>4</td><td>1</td><td>8</td><td>6</td></tr></table> <p>In the 6th iteration, the index[6] data is used as the inserted data , and compared with previous data. There are three greater data than the inserted data, then the three data must be shifted one place rightside and the inserted data shifted to the position where the three data before moved.</p>	0	2	3	5	7	9	4	1	8	6
0	2	3	5	7	9	4	1	8	6		
1	<table><tr><td>0</td><td>2</td><td>3</td><td>4</td><td>5</td><td>7</td><td>9</td><td>1</td><td>8</td><td>6</td></tr></table> <p>In the 7th iteration, the index[7] data used as the inserted data, then compared with previous data. There are six greater data than the inserted data, then the six data must be shifted one place rightside and the inserted data moved into place where the six data before moved</p>	0	2	3	4	5	7	9	1	8	6
0	2	3	4	5	7	9	1	8	6		
8	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>7</td><td>9</td><td>8</td><td>6</td></tr></table> <p>In the 8th iteration, the index[8] data is used as the inserted data, and compared with previous data. There is one greater data than the inserted data, so the greater data must be shifted one place rightside and the inserted data shifted to the place where the greater data before shifted.</p>	0	1	2	3	4	5	7	9	8	6
0	1	2	3	4	5	7	9	8	6		

6	0	1	2	3	4	5	7	8	9	6
	In the 9th iteration, the index[9] data used as the inserted data, and then compared with previous data. There are three greater data than the inserted data, then the three data must be shifted one place rightside and the inserted data moved into position where the three data before moved.									
Final Result	0	1	2	3	4	5	6	7	8	9

If the above steps is implemented in C++ will be as below.

```

1  #include<iostream>
2  using namespace std;
3  int main() {
4      int data[]={5,3,7,2,0,9,4,1,8,6};
5      int i,dataSize=sizeof(data)/sizeof(data[0]),temp;
6      for(int j=1;j<dataSize;j++){
7          i=j-1;
8          temp=data[j];
9          while(data[i]>temp&& i>=0){
10             data[i+1]=data[i];
11             i--;
12         }
13         data[i+1]=temp;
14     }
15     for(int j=0;j<dataSize;j++){
16         cout<<j<<" ";
17     }
18     return 0;
19 }
```

### 8.2.2. Selection Method (*Selection Sort*)

Selection method is a sorting method that searches for smallest or largest values depended on the ascending or descending order, and then placed into the leftmost place, then search again for the second smallest or largest value along the number of array elements subtracted by one.

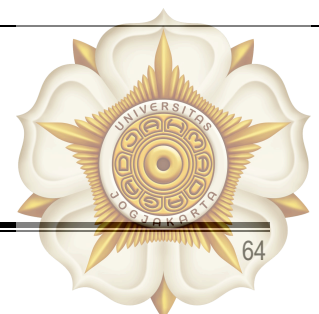
Once found, the second element is exchanged with the found value, and so on. For example, if there is an array that contains the following rates(value).

5	3	7	2	0	9	4	1	8	6
---	---	---	---	---	---	---	---	---	---

If the above data will be sorted in ascending order using the selection sorting, so the process is as follows.



Smallest Value	Sorting Result																			
0	<table><tr><td>5</td><td>3</td><td>7</td><td>2</td><td>0</td><td>9</td><td>4</td><td>1</td><td>8</td><td>6</td></tr></table> <p>In the 1st iteration, search the smallest value of the index array elements 0-9 and found a value 0 as the smallest value, then the place(element) whose the value is 0 exchanged with the element of index 0.</p>										5	3	7	2	0	9	4	1	8	6
5	3	7	2	0	9	4	1	8	6											
1	<table><tr><td>0</td><td>3</td><td>7</td><td>2</td><td>5</td><td>9</td><td>4</td><td>1</td><td>8</td><td>6</td></tr></table> <p>In the 2nd iteration, search the smallest value of the index array elements 1-9 and found a value 1 as the smallest value, then the place(element) whose the value is 1 exchanged with element of index 1</p>										0	3	7	2	5	9	4	1	8	6
0	3	7	2	5	9	4	1	8	6											
2	<table><tr><td>0</td><td>1</td><td>7</td><td>2</td><td>5</td><td>9</td><td>4</td><td>3</td><td>8</td><td>6</td></tr></table> <p>In the 3rd iteration, search the smallest value of the index array elements 2-9 and found a value 2 as the smallest value, then the place whose the value is 2 exchanged with the element of index 2.</p>										0	1	7	2	5	9	4	3	8	6
0	1	7	2	5	9	4	3	8	6											
3	<table><tr><td>0</td><td>1</td><td>2</td><td>7</td><td>5</td><td>9</td><td>4</td><td>3</td><td>8</td><td>6</td></tr></table> <p>In the 4th iteration, search the smallest value of the index array elements 3-9 and found a value 3 as the smallest value, then the place whose the value is 3 exchanged with element of index 3.</p>										0	1	2	7	5	9	4	3	8	6
0	1	2	7	5	9	4	3	8	6											
4	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>5</td><td>9</td><td>4</td><td>7</td><td>8</td><td>6</td></tr></table> <p>In the 5th iteration, search the smallest value of the index array elements 4-9 and found a value 4 as the smallest value, then the place whose the value is 4 exchanged with the element of index 4</p>										0	1	2	3	5	9	4	7	8	6
0	1	2	3	5	9	4	7	8	6											
5	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>9</td><td>5</td><td>7</td><td>8</td><td>6</td></tr></table> <p>In the 6th iteration, search the smallest value of the index array elements 5-9 and found a value 5 as the smallest value, then the place whose the value is 5 exchanged with element of index 5.</p>										0	1	2	3	4	9	5	7	8	6
0	1	2	3	4	9	5	7	8	6											
6	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>9</td><td>7</td><td>8</td><td>6</td></tr></table> <p>In the 7th iteration, search the smallest value of the index array elements 6-9 and found the value 6 as the smallest value, then the place whose the value is 6 exchanged with element of index 6.</p>										0	1	2	3	4	5	9	7	8	6
0	1	2	3	4	5	9	7	8	6											
7	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table> <p>In the 8th iteration, search the smallest value of the index array elements 7-9 and found the value 7 as the smallest value. Element whose the value is 7 have been in proper place then it no need to be exchanged.</p>										0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9											



8	0	1	2	3	4	5	6	7	8	9
In the 9th iteration, search the smallest value of the index array elements 8-9 and found a value 8 as the smallest value. Element whose the value is 8 have been in proper place then it no need to be exchanged.										
Final Result	0	1	2	3	4	5	6	7	8	9

The above steps can be implemented in C ++ as follows.

```

1  #include<iostream>
2  using namespace std;
3  int main() {
4      int data[]={5,3,7,2,0,9,4,1,8,6};
5      int dataSize=sizeof(data)/sizeof(data[0]),min,temp;
6      for(int i=0;i<dataSize-1;i++){
7          min=i;
8          for(int j=i+1;j<dataSize;j++){
9              if(data[j]<data[min]){
10                 min=j;
11             }
12         }
13         temp=data[i];
14         data[i]=data[min];
15         data[min]=temp;
16     }
17     for(int i=0;i<dataSize;i++){
18         cout<<i<<" ";
19     }
20     return 0;
21 }

```

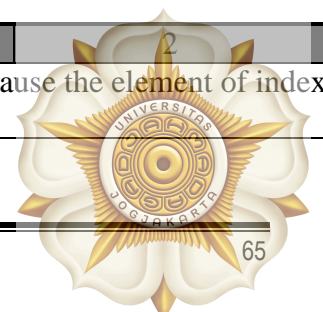
### 8.2.3. Bubble Method (Bubble Sort)

The method of bubbles is sorting method that exchange two-element continuously until sorting is complete. For example, if there is an array that contains the following figures.

5	3	7	2
---	---	---	---

If the above data will be sorted in ascending order using the bubble then the process is as follows.

5	3	7	2
1st iteration compares the element of index 0 with element of index 1, because the element of index 0 is greater than element of index 1 so there is an exchange.			
3	5	7	2
2nd iteration compares elements of index 1 and element of index 2, because the element index 1 is not greater than the element index 2 then there is no exchange.			
3	5	7	2
3rd iteration compares elements of index 2 with element of index 3, because the element of index 2 is greater, then do the exchange.			



3	5	2	7
4th iteration compares elements of index 0 and element of index 1, because the element of index 0 is not greater, then there is no exchange.			
3	5	2	7
5th iteration compares elements of index 1 with element of index 2, because the element index 1 is greater , then do the exchange.			
3	2	5	7
6th iteration compares elements of index 2 with element of index 3, because the element of index 2 is not greater than the element of index 3 then noting to be exchange.			
3	2	5	7
7th iteration compares elements of index 0 with element of index 1, because the element of index 0 is greater, then do the exchange			
2	3	5	7
8th iteration compares elements of index 1 with element of index 2, because the element index 1 is not greater then there is no exchange.			
2	3	5	7
9th iteration compares elements of index 2 and element of index 3, because the element of index 2 is not greater,then there is no exchange.			

If the above steps is implemented in C++ will be as below.

```

1  #include<iostream>
2  using namespace std;
3  int main(){
4      int data[]={5,3,7,2,0,9,4,1,8,6};
5      int dataSize=sizeof(data)/sizeof(data[0]),temp;
6      for(int i=0;i<dataSize-1;i++){
7          for(int j=0;j<dataSize-1;j++){
8              if(data[j]>data[j+1]){
9                  temp=data[j];
10                 data[j]=data[j+1];
11                 data[j+1]=temp;
12             }
13         }
14     }
15     for(int i=0;i<dataSize;i++){
16         cout<<i<<" ";
17     }
18     return 0;
19 }

```

#### 8.2.4. MERGE SORT

Merge sort is a advance sorting method, it has same way to sort array just like quick sort.

Merge sort use divide and conquer concept which divide S data into two groups which are S1 and S2 which is disjoint each other. Dividing process was done recursively until data cannot divided anymore, merging process was done by merge between sub arrays by considering order (ascending/ descending). Combining process was done until whole data is



combined and ordered as our wish. Generally, merge sort can be implemented recursively. Recursive function is a function which on its implementation will call itself into it and will end when some condition was triggered.

Here's how merge sort works:

- 1. Divide** by finding the number Q of the position midway between P and R.
- 2. Conquer** by recursively sorting the subarrays in each of the two subproblems created by the divide step.
- 3. Combine** by merging the two sorted subarrays back into the single sorted subarray.

For example, there is an array which contains some value.

14          7          3          12          9          11          6          2



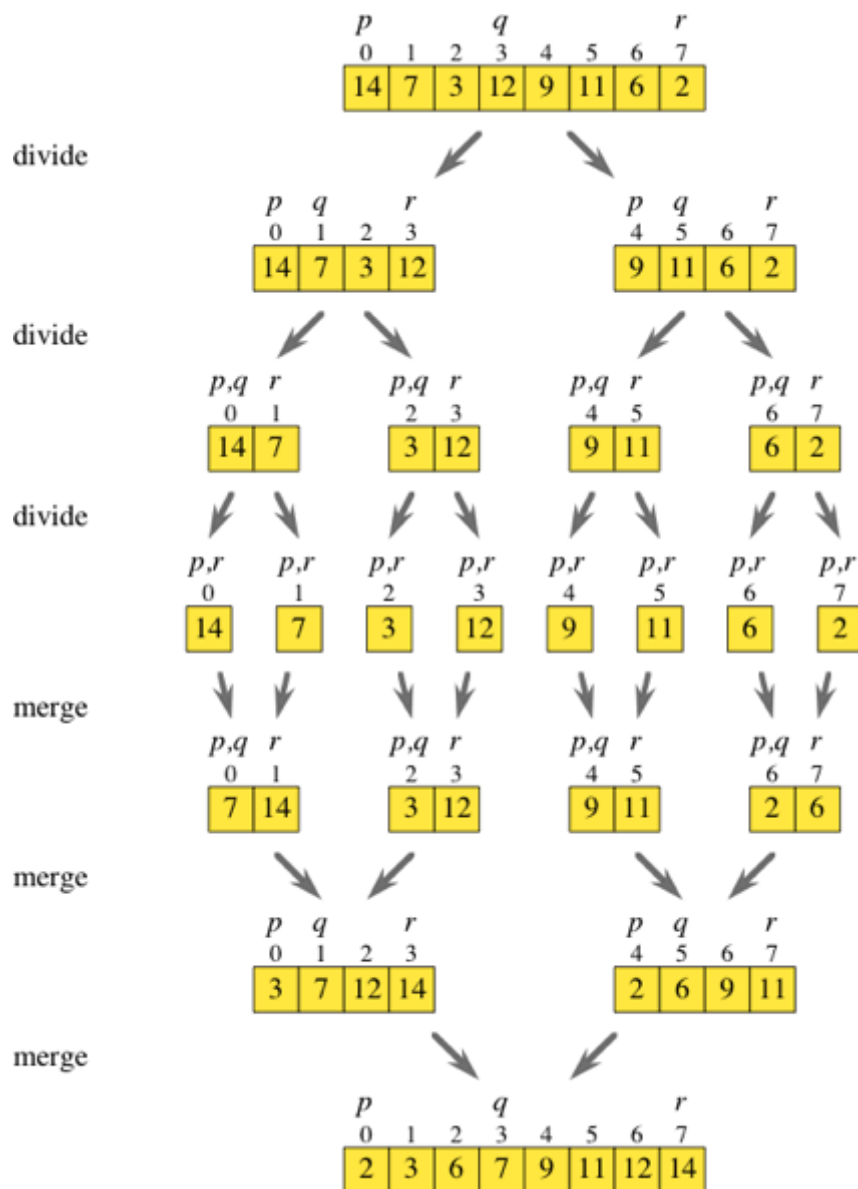


Figure xxx example for sorting array using merge sort



```

1  #include <iostream>
2  #include<stdio.h>
3
4  using namespace std;
5
6  void merge_sort(int low,int high)
7  {
8      int mid;
9      if(low<high)
10     {
11         mid = low + (high-low)/2;
12         merge_sort(low,mid);
13         merge_sort(mid+1,high);
14         merge(low,mid,high);
15     }
16 }
17

```

Figure implementation of merge sort on c++

### 8.2.5. QUICK SORT

Quicksort is an divider algorithm for comparing an element called pivot with any other element and organize it as possible until other elements on its left are lesser and the rest of element on the right side are bigger and will generate 2 sub lists on both side of pivot. And will repeatedly until there is not anymore sub list.

Here's how quick sort works:

1. Divide by choosing any element in the subarray array[p..r]. Call this element the pivot. Rearrange the elements in array[p..r] so that all other elements in array[p..r] that are less than or equal to the pivot are to its left and all elements in array[p..r] are to the pivot's right. We call this procedure partitioning. At this point, it doesn't matter what order the elements to the left of the pivot are in relative to each other, and the same holds for the elements to the right of the pivot. We just care that each element is somewhere on the correct side of the pivot.
2. Conquer by recursively sorting the subarrays array[p..q-1] (all elements to the left of the pivot, which must be less than or equal to the pivot) and array[q+1..r] (all elements to the right of the pivot, which must be greater than the pivot).

For example, there is an array which contains some value.

9      7      5      11      12      2      14      3      10      6



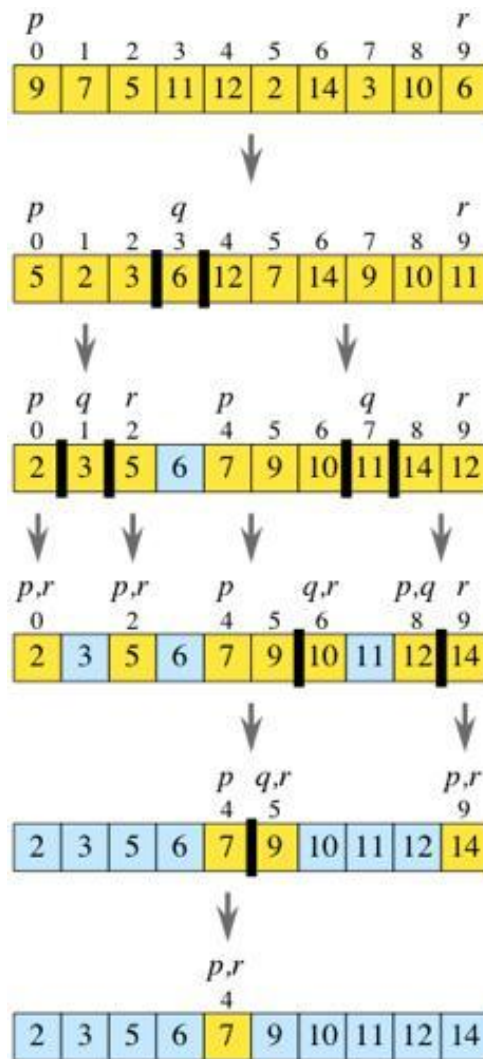


Figure xxx example for sorting array using quick sort



```

1  #include <iostream>
2  #include<stdio.h>
3
4  using namespace std;
5
6  void QuickSort(int * Data, int a,int b )
7  {
8      int al,b1,pivot;
9      al = a; b1 = b;
10     pivot = Data[(a+b) / 2];
11     while (!(al>b1))
12     {
13         while(Data[al] < pivot) al++;
14         while(Data[b1] > pivot) b1--;
15         if (al <= b1)
16         {
17             Tukar(Data[al],Data[b1]);
18             al++; b1--;
19         }
20     }
21     if (a<b1) QuickSort(Data,a,b1);
22     if (al<b) QuickSort(Data,al,b);
23 }
24

```

Figure implementation of Quicksort on c++

### 8.2.6. Search

Search data on a set of data is a very important process in real life. As well as sorting, searches can also be performed with multiple search methods such as successive search method (sequential search) and the method of divided by two (binary search) that will be discussed..

#### Successive Search (*Sequential Search*)

Successive search can be performed on set of sorted data or unsorted data. Successive search done by tracing the data one by one and then matched with the wanted data, if not the same then the search continues, if the same data found then the search is stopped. If there are data as follows.

Identification number	Name	Value
10101	Adi	64.75
10103	Budi	75.11
10105	Charli	84.63
10102	Dodi	77.07
10104	Edi	66.70

Charli's value will be searched in terms of the data that belongs to the Charli is identification number "10105". If the successive search implemented in C ++ will be as follows..



```

1  #include<iostream>
2  using namespace std;
3  int main() {
4      bool found=false;
5      char* nama={"Nana","Rudi","Dea","Ihsan","Tiara"};
6      char* nomorInduk={"13507701","13507702","13507703","13507704","13507705"};
7      char* query="13507703";
8      float nilai[]={64.75,75.11,84.63,77.07,66.70};
9      for(int i=0;i<5;i++){
10         if(nomorInduk[i]==query){
11             cout<<nama[i]<<","<<nomorInduk[i]<<","<<nilai[i]<<endl;
12             found=true;
13         }
14     }
15     if(!found){
16         cout<<"Tidak ditemukan.";
17     }
18     return 0;
19 }

```

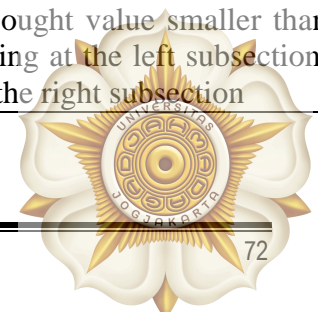
### Divided in two Search (*Binary Search*)

Searches of Divided by two, or binary search, can only be performed on the data that has been (sorted)ordered. For example, if there is an array that contains the following figures..

5	3	7	2	0	9	4	1	8	6
---	---	---	---	---	---	---	---	---	---

If the sought number is 8 then the process is as follows.

Steps	Description										
1	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table> <p>3. The table is still considered as one entity table that contains the data that has been ordered.</p>	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9		
2	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table> <p>4. The table is divided into two</p> <p>5. Check the rightmost value of the left subsection and the leftmost value of right subsection..</p> <p>6. If the sought value is greater or equal to the leftmost value of the right subsection, search for the right subsection. If the sought value smaller than the rightmost value of the left subsection then looking at the left subsection, because the search is for 8 then should be sought in the right subsection</p>	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9		
3	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table> <p>7. Right subsection divided into two parts..</p> <p>8. Check the rightmost value of the left subsection and the leftmost value of right subsection..</p> <p>9. If the sought value is greater or equal to the leftmost value of the right subsection, search for the right subsection. If the sought value smaller than the rightmost value of the left subsection then looking at the left subsection, because the search is for 8 then should be sought in the right subsection</p>	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9		



4	0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---	---

10. The table is divided into two parts

11. Check the rightmost value of the left subsection and the leftmost value of right subsection.

12. If the sought value is greater or equal to the leftmost value of the right subsection, search for the right subsection. If the sought value smaller than the rightmost value of the left subsection then looking at the left subsection, because the search is for 8 then should be sought in the right subsection

The above steps if written in C ++ will be as follows.

```

1  #include<iostream>
2  using namespace std;
3  int main() {
4      bool found=false;
5      int data[]={0,1,2,3,4,5,6,7,8,9};
6      int i=0,j=sizeof(data)/sizeof(data[0]),k,query=7;
7      while(!found&&i<=j){
8          k=(i+j)/2;
9          if(data[k]<query){
10             i=k+1;
11         }else if(data[k]==query){
12             found=true;
13         }else{
14             j=k-1;
15         }
16     }
17     if(!found){
18         cout<<"Tidak ditemukan.";
19     }else{
20         cout<<"Ditemukan.";
21     }
22     return 0;
23 }
```



### 8.3. Activity

Student do the exercise

### 8.4. Exercise

Note the following student data.

NISN	Nama	Value
9960312699	Handi Ramadhan	90
9963959682	Rio Alfandra	55
9950310962	Ronaldo Valentino Uneputty	80
9970272750	Achmad Yaumil Fadjri R.	60
9970293945	Alivia Rahma Pramesti	70
9952382180	Ari Lutfianto	65
9965653989	Arief Budiman	60

1. Sort the data in descending by:

- NISN
- Value

using insertion sort, selection sort and bubbles sort.

2. Look for data that has NISN 9950310962, then show the value using the binary search.

3. Change the name of the data that has a value of 60 to Joko. Take advantage of sequential search methods.





## CHAPTER IX POINTER

### 9.1 Learning Objectives

1. Describe concept of pointer
2. Understand the different between pointer and other data structures

#### Competence

1. Students understand concept of pointer
2. Students can implementating pointer into link list problem

### 9.2 Theory

The Pointer concept is actually quite simple . the pointer actually contains the address of the data, not the data as variables that you know.

Each byte in the computer's memory has an address. A variable is stored in this memory. But of course the programmer does not need to mention the address of a variable explicitly. By the time the program is loaded in memory, the variable will be placed automatically on a specific address..

A pointer in the program is intended to refer to a memory address. For example, if **pint** is a pointer, and **vint** is a variable that is located in **0xffff2** memory address, pint pointer can be set to point to (to refer to) the variable **vint**.

Address of a variable can be determined easily, by adding the address operator or (&) symbol in front of the nama\_variabel. By submitting to **cout**, the address of a variable will be displayed to the screen. Here is a sample program to display the memory address of a variable.



```

#include <iostream>
using namespace std;

int main() {
    int bil1 = 5;
    float bil2 = 7.5;
    double bil3 = 17.777;

    cout<<"Isi variabel : "<<endl;
    cout<<"bil1 = "<<bil1<<endl;
    cout<<"bil2 = "<<bil2<<endl;
    cout<<"bil3 = "<<bil3<<endl;

    cout<<endl;
    cout<<"Alamat variabel : "<<endl;
    cout<<"bil1 = "<<&bil1<<endl;
    cout<<"bil2 = "<<&bil2<<endl;
    cout<<"bil3 = "<<&bil3<<endl;
    return 0;
}

```

Keep in mind, the 0x notation states hexadecimal in the program result. In C++, (&) mark is also used as reference (an alias of a variable).

As previously mentioned, The pointer is a variable that can store memory addresses. In the previous chapter we are used to use variables, but the variable contains a value not address.

Here's a pointer declaration form..

```

type_data *nama_variabel;

```

**type\_data** can be any data type as the non-pointer variable. The **nama\_variabel** is a variable pointer. In the purpose of that the pointer points to another variable, that must first be filled with the address of the variables to be appointed(referred). For example, there is a definition as follows:

```

int vint = 55;
int *pint;

```

**Pint** variables can be set to point to **vint** the following way::

```

pint = &vint;

```

The above statement means: "**pint** is filled with the address of **vint**".



Here is an example of a complete program..

```
#include <iostream>
using namespace std;

int main() {
    int vint = 55;
    int *pint;
    pint = &vint;    //pointer menunjuk ke vint
    cout<<"Alamat vint = "<<&vint<<endl;
    cout<<"Isi pint = "<<*pint<<endl;
    return 0;
}
```

Based on the above examples, the value of **vint** can be accessed through a **pint** after the following statement is executed.

```
pint = &vint;
```

The way to access it is to involve "indirect" operator whose symbol is \* and positioned in front of the name of the pointer variable. Example:

```
*pint;
```

Means "the value which referred(pointed) by the **pint**".

The above program shows accessing the **vint** can be done through \* **pint**. The following example shows the contents of vint changed via \* **pint**.

```
#include <iostream>
using namespace std;

int main() {
    int vint = 55;
    int *pint;
    pint = &vint;    //pointer menunjuk ke vint

    cout<<"Isi vint semula = "<<vint<<endl;
    *pint = 77;
    cout<<"Isi vint sekarang = "<<vint<<endl;
    return 0;
}
```



## Void Pointer

In the above examples program have been shown the pointer that points to a specific data type. Actually, it is also possible to create a pointer without type. This way be done by allocate the keyword **'void'** in the type of pointer or in (leftside of **\* notation**). Example:

```
void *ptr;
```

A void pointer as the above example can be used to point to any data type.

```
#include <iostream>
using namespace std;

int main() {
    int bil1 = 55;
    float bil2 = 33.45;
    void *ptr;

    ptr = &bil1; //bisa menunjuk ke int
    cout<<"Nilai yang ditunjuk ptr : "<<*(int*)ptr<<endl;

    ptr = &bil2; //bisa menunjuk ke float
    cout<<"Nilai yang ditunjuk ptr : "<<*(float*)ptr<<endl;
    return 0;
}
```

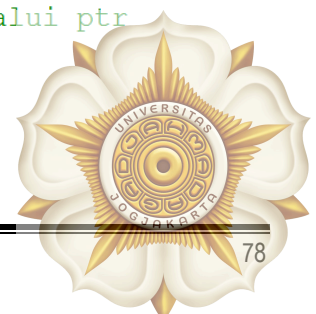
In the above program, the following forms need to be given:

```
*(int*)ptr
*(float*)ptr
```

This is due to C++ does not know the purpose of \* ptr, considering ptr is a void pointer. This method is not only applicable if we intend to take the value pointed by the ptr but also when we change it. As an example of the following program:

```
#include <iostream>
using namespace std;

int main() {
    int bilangan = 55;
    void *ptr;
    ptr = &bilangan;
    cout<<"bilangan semula : "<<bilangan<<endl;
    *(int*)ptr = 77; //mengubah nilai bilangan melalui ptr
    cout<<"bilangan sekarang : "<<bilangan<<endl;
    return 0;
}
```



## REFERENCE

Binanto, Iwan, 2003, Pemrograman C++ di Linux, Penerbit Andi, Jogjakarta.

Binanto, Iwan, 2004, Lebih Lanjut dengan Pemrograman C++ di Linux, Penerbit Andi, Jogjakarta.

Kadir, Abdul, 2004, Pemrograman Visual C++, Penerbit Andi, Jogjakarta.

Shalahuddin, M. dan A. S., Rosa, 2009, *Belajar Pemrograman dengan Bahasa C++ dan Java*, Penerbit Informatika, Bandung.

Java 2 TM Fourth Edition, McGraw Hill, New York

[www.juicy.com/tutorial/c++](http://www.juicy.com/tutorial/c++)

