

CHAPTER 2

BASIC ELEMENTS OF PROGRAMMING LANGUAGE

Chapter's Outcome

At the end of this chapter, student should be able to:

- a) Identify the basic elements needed in writing a program
- b) Understand and apply the input/output statement
- c) Justify the process of error handling
- d) Write a simple program

Chapter's Outline

- 1) Identifier, variable, constants and reserved words
- 2) Rules for naming identifiers
- 3) Declaring data variables and constants
- 4) Basic data types –integer, float, double, bool, char , string
- 5) Arithmetic operators – addition, subtraction, multiplication, division, modulus
- 6) Assignment expression
- 7) Precedence & associativity
- 8) Assignment statement
- 9) Introduce some mathematical library functions
- 10) Input/output statement
- 11) Formatting output
- 12) Programming process, debugging and error handling
- 13) Three types of control structure – sequential, selection, repetition/looping

What is Identifiers?

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world";

    return 0;
}
```

a)By referring the above simple program, there are several identifiers are detected. The words include, iostream, int, main, cout, {, } are considered as identifiers.

b)So another word is there are many identifiers will be used when you write programs. Identifier is also related with variable and constant. An identifier in C++ is case sensitive. For example, the identifier Sum is not the same identifier as sum or SUM.

c)For more information about identifiers please refer :

<http://oakroadsystems.com/tech/c-prefdef.htm#Table>

http://www.acm.uiuc.edu/webmonkeys/book/c_guide/index.html

What is Header File?

Each program must contain the header file or library. Eg : [iostream.h](#), [iomanip.h](#), [math.h](#), [conio.h](#), [string.h](#). If the header file is not included, the program will contain many errors.

New C++ Name	Old C/C++ Name	Use and functions
iostream	iostream.h	Defines insertion (<<) and extraction (>>) operators and creates the global stream objects like cin and cout. See also, Stream Member Functions
iomanip	iomanip.h	Provides a variety of stream formatting and manipulation tools. See also, Stream Manipulators section.
fstream	fstream.h	Required for file I/O operations. See also, Stream Member Functions .
cmath	math.h	Provides a wide range of special math functions. These include the trigonometric functions (with angles expressed in radians), exp(double x), log(double x), log10(double x), pow(double base, double power), sqrt(double x), fabs(double x) and fmod(double numerator, double denominator).
cstdlib	stdlib.h	Miscellaneous stuff. Including the void srand(int seed) and int rand() pseudo-random random number functions and the int system(const char command[]) system command function. The exit(int exit_code) does not seem to be supported in the MSVC++ 6.0 version of
cassert	assert.h	Provides the assert error handling mechanism. This has largely been replaced by the exception mechanism in bigger programs, but is still useful in smaller ones.
string	None	Provides the string class. Note that <string> is completely different than the old <string.h> library (now <cstring.h>). See also, String Class Member Functions and Operators
vector	vector.h	Provides the Standard Template Library (STL) implementation of a 1-dimensional, random access sequence of items. Generally replaces the use of 1-dimensional C/C++ arrays. See also, Vector Class Member Functions and Operators and the MSVC++ 6.0 <valarray> include file.
ctime	time.h	Types and functions associated with calendar and time operations. Includes both processor and actual time and date functions.
complex	None	Provides a template class for storing and manipulating complex numbers.

[Return to Table of Contents](#)

What are Variables?

- a) Variables also referred to as *"identifiers"* are named memory locations that have a type, such as an integer or character, and consequently, a size, which is inherited from their type.
- b) Since variables are types, they have a set of operations that can be used to change or manipulate them.
- c) Variable is a location in memory that holds changeable data.
- d) An identifier name can be made up of only the following:
 - Letters - "a" through "z", and "A" through "Z".
 - Numbers - 0 through 9.
 - The underscore character "_".

What are Variables? (cont...)

- e) Identifiers must begin with a letter or an underscore. However, some identifiers beginning with an underscore have special meanings, so generally user-defined identifiers begin with a letter.
- f) There are some reserved words in C++, so the programmer should be cautious in his choice of identifier names. For example, "int" is a poor choice, since that is a reserved word indicating the *type* of the variable.
- g) Program won't be interesting, if never involve declaration of variables. There are four types of data types int, char and float and double.
- h) Samples of variable:

Student, employee_number, AGE, _marks1, mARK2

Rules of naming/declaring variable

- 1) Start with alphabetic or underscore
- 2) Don't declare the same variable more than once.
- 3) Variable cannot start with number. Eg: `1marks`, `4_age`.
- 4) A variable may NOT contain blanks (**no space between variable's name**)
- 5) Cannot use the reserved word to declare variable. Eg : `asm`, `auto`, `break`, `case`, `catch`, `char`, `class`, `const`, `continue`, `default`, `delete`, `do`, `double`, `else`, `enum`, `extern`, `float`, `for`, `friend`, `goto`, `if`, `inlineint`, `long`, `new`, `operator`, `private`, `protected`, `public`, `register`, `return`, `short`, `signed`, `sizeof`, `static`, `struct`, `switch`, `template`, `this`, `throw`, `try`, `typedef`, `union`, `unsigned`, `virtual`, `void`, `volatile`, `wchar_t`, `while`,

Rules of naming/declaring variable (cont...)

6) Suggestion when you declare variables.

- Meaningful
- Understandable
- Avoid ambiguity
- Not too long but looks simple

Exercise on naming variables:

Identify either the variable's name is **Valid** or **Invalid**:

Variable	Valid/Invalid
TotalNum	
int	
VOID	
sum Num	
aVeraGe32	
#area	
Discount*price	
plus987abc	

Exercise on naming variables:

Answers:

Variable	Valid/Invalid
TotalNum	Valid
int	Invalid
VOID	Valid
sum Num	Invalid
aVeraGe32	Valid
#area	Invalid
Discount*price	Invalid
plus987abc	Valid

Variable Types

Refer to any text for a complete set of variable types. Here is a short list of commonly used types.

<code>char</code>	A 1 byte integer usually used to store a character.
<code>short</code>	A 2 byte integer.
<code>int</code>	A 4 byte integer.
<code>long</code>	4 or 8 byte integer (compiler dependent).
<code>float</code>	Single precision floating point number. (7 digits or more - compiler dependent)
<code>double</code>	Double precision floating point number. (15 digits or more - compiler dependent)
<code>bool</code>	Boolean literal. Values are true (numerically=1) or false (numerically=0).
<code>string</code>	A variable number of characters. Note: string is actually a C++ <u>class</u> . In order to use this data type, you need to add the statement: <code>#include <string></code> to the start of your program.

Declaring a variable

Declaring a variable starts with data type followed by the name of variable and a semicolon.

Syntax:

```
type    variable_name;
```

Example:

```
int age;           //Declare age as integer.
float sum;         //Declare sum as float.
double area;       //Declare area as double
char name[25];     //Declare name as character.
string address=""; //Declare address as string.
char gender;       //Declare gender as character.
```

Sample program using **integer** data type:

```
#include <iostream>
using namespace std;
int main()
{
    int value1, value2;

    value1 = 4;
    value2 = 5;

    cout << value1 << value2;

    return 0;
}
```

"This program shows the usage of integer data type. Two variables have been declared value1 and value2. Both data type is integer. Means both variables accept numbers only.

"If we relates to the concepts of memory management, there are two addresses have been used to store the number 4 and 5 at value1 and value2 respectively.

Eg:

value1

4

value2

5

Sample program using **float** data type:

```
#include <iostream>

using namespace std;

int main()
{
    float pai;
    pai = 3.14;
    cout << pai;

    return 0;
}
```

- pai is a variable which the data type is float.
- Float accepts until 7 decimal points.
- Others examples of variable using float are currency, GPA, CGPA and etc.

Sample program using **double** data type:

```
#include <iostream>

using namespace std;

int main()
{
    double result;
    result = 3.263772838;
    cout << result;

    return 0;
}
```

- This program shows that the variable use double data type.
- Double supports more than 7 decimal points up to 14.

Sample program using **char** data type:

```
#include <iostream>

using namespace std;

#include <string.h>

int main()
{
    char code;
    char symbol;
    char name[30];
    code = 'b';
    symbol = '+';
    cout << code;
    cout << symbol;
    cout << strcpy(name, "Don't Panic");

    return 0;
}
```

•They are two types of char, either you enter an alphabet/symbol or string.

•If it is a string then the function strcpy() should be used.

•Don't forget to declare the header file string.h.

Sample program using **string** data type:

```
#include <iostream>

using namespace std;

#include <string.h>

int main()
{
    string courseCode;

    cout << "Please enter your course code: ";
    cin >> ws;
    getline (cin, courseCode);
    cout << "Course code registered is " <<courseCode<<endl;

    return 0;
}
```

- A string is a collection of characters, for example name, address, matrix number or phone number.
- String in C++ always enclosed in double quotes (" ").
- Don't forget to declare the header file string.h.

What is Constant?

- 1) Constants provide a way to define **a variable which cannot be modified** by any other part in the code.
- 2) Constants can be defined by placing the keyword **const** in front of any variable declaration.
- 3) May not have its value changed while the programming is running.
- 4) Constant are usually define in capital letter.

Example:

```
const double PAI = 3.142;
```

Basic Arithmetic Operators

Basic arithmetic's operators are:

Arithmetic Operators in C++	Arithmetic Operation	Example
+	Addition	$57.5 + 6 = 73.5$, or $10 + 8 = 18$
-	Subtraction	$100 - 19 = 81$, or $66.4 - 15.5 = 50.9$
*	Multiplication	$7 * 15 = 105$, or $30.2 * 7 = 211.4$
/	Division	$19 / 2 = 9$, or $19.0 / 2 = 9.5$
%	Modulus (remainder)	$15 \% 3 = 0$ or $17 \% 3 = 2$ Never use the modulus with floating-point (decimal) values.

Arithmetic Expression

For example if you are given the following variables.

`Value1 = 5`

`Value2 = 10`

`Value3 = 7`

a)Operator '+' is used to get the addition value of variables.

`Value1+Value2 = 15`

b)Operator '-' is used for subtraction of variables.

`Value3-Value1 = 2`

c)Operator '*' is used to multiply variables.

`Value1*Value2*Value3 = 350`

•Operator '/' is used to get the division value of variables.

`Value2/Value1 = 2`

•Operator '%' is used to get the balance of division result.

`Value2%Value3 = 3`

Exercise on Arithmetic Expression

Convert the following mathematics expression into arithmetic expression (in C++).

i) $a = b + cd - ef$

ii) $c = a + \frac{b}{de} - fg$

iii) $a = \sqrt{x + y^2 - \frac{z}{2}}$

Exercise on Arithmetic Expression

Write the equivalent C++ assignment statement for the following algebraic equations.

i)
$$F = k \left(\frac{M_1 M_2}{d^2} \right)$$

ii)
$$T = 2\pi \sqrt{\frac{m}{k}}$$

Precedence & Associativity

Parenthesis is used in C++ expressions in much the same manner as in algebraic expressions.

Example:

`a * (b+c)`

There are several rules that you need to know:

- a) Operators in expressions contained within pairs of parenthesis are evaluated first.
- b) *, / and % are applied next.
- c) + and – operations are applied last.

Precedence & Associativity (cont...)

Example:

Evaluate the following statements. **Assume that all variables are integer.**

int n = 9, m = 16, a = 2, c = 7, X;

a) $X = 1 - (1 - (1 - (1 - n)))$

you must solved in the bracket first then go to outer bracket.

b) $X = m / n + n \% m$

c) $X = -m + \text{sqrt}(n * n - a * m) / 2 * a$

Precedence & Associativity (cont...)

Example:

Evaluate the following statements. Assume that all variables are integer.

$n = 9, m = 16, a = 2, c = 7$

a) $1-(1-(1-(1-n))) = 9$

you must solved in the bracket first then go to outer bracket.

b) $m / n + n \% m$

$m / n = 1$ then $n \% m = 9$, the final answer is 10

c) $- m + \text{sqrt} (n * n - a * m) / 2 * a$

$\text{sqrt} (n * n - a * m) = 7$

then divide by 2 = 3

then $* a = 6$

$- m + 6 = -10$

Exercise on Precedence & Associativity

What is the output of each of the following arithmetic expressions?

i) $22 / 6 * 3$

ii) $14 + 7 \% 2 - 5$

iii) $13.0 + 9 / 2.0$

Writing Program (Exercise)

Write a program that ask the user to enter two integers. Then, compute the addition, subtraction, multiplication and division of the two integers. The program will display the output of all the mathematical operations above.

```
Enter first integer: 40  
Enter second integer: 8
```

```
-----  
Mathematical Operations  
-----
```

```
40 + 8 = 48  
40 - 8 = 32  
40 x 8 = 320  
40 / 8 = 5  
-----
```

Commenting the Program

- 1) Comment is very important in any program.
- 2) The purpose of commenting the program is to:
 - make us easier to refer and
 - understand the function of the program.
- 3) Normally the program will consists like author's name, date program created, purpose, etc.

Example:

```
//Author : Aminah Bt Abu Bakar  
//Date : 7 May 2002  
//Purpose : to calculate the total of two numbers
```

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int a, b;  
    a=5;  
    b=6;  
    cout << a+b;  
    return 0;  
}
```

Input and Output Statements

Among the statements in sequential structure in C++:

Output statement : `cout <<`

Input statement : `cin >>`

Assignment statement : `=`

Assignment Statement

- 1) To assign a value to variable without keying in the data.
- 2) The value has been assigned in the program.

Example:

```
mark = 3; //assigning an integer value
grade = 'A'; //assigning a character value
strcpy(name, "Aziz"); //assigning a string value
total = 33.5; //assigning a double value
courseCode = "CSC128"; //assigning a string value
```

Assignment Statement (cont...)

- 1) A variable can store only one value at a time.

Example:

```
number = 12.7;  
number = 35.2;
```

- The value **12.7** will be assigned to variable **number** and then the value **35.2** will be assigned to variable **number**. The value **12.7** will be overwrite and **number** would retain the latest value **35.2**.

Assignment Statement (cont...)

- 2) Two variables can also store the same value.

Example:

```
int number1, number2;
```

```
number1 = 12.7;
```

```
number2 = number1;
```

- The value **12.7** will be assigned to variable **number1**.
- Next, the variable **number2** takes the value of variable **number1 = 12.7**.
- Now, both variable **number1** and **number2** store the same value which is **12.7**

Assignment Statement (cont...)

- 3) A modified variable can also store the final result into the origin variable.

Example:

```
int count;
```

```
count = count + 10;
```

- Retrieve from memory the contents of memory cell **count**.
- Next, add ten to the value of variable **count**.
- Finally, store the result in the memory cell named **count**.

Increment and Decrement Operator (Postfix & Prefix)

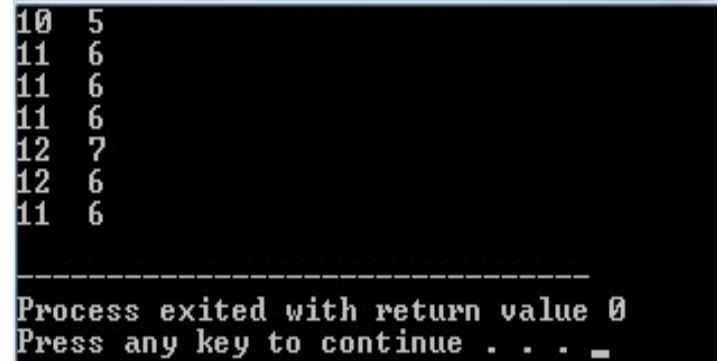
Operator Name	Operator	Description
Increment postfix	<code>a++</code>	<code>a = a + 1</code>
Increment prefix	<code>++a</code>	<code>a = a + 1</code>
Decrement postfix	<code>a--</code>	<code>a = a - 1</code>
Decrement prefix	<code>--a</code>	<code>a = a - 1</code>

Sample program using **Postfix** and **Prefix** operator:

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

int main()
{
    int x = 10, y = 5;
    cout << x << " " << y << endl;
    cout << ++x << " " << ++y << endl; // prefix
    cout << x << " " << y << endl;
    cout << x++ << " " << y++ << endl; // postfix
    cout << x << " " << y << endl;
    cout << x-- << " " << --y << endl; // postfix & prefix
    cout << x << " " << y << endl;

    return 0;
}
```



```
10 5
11 6
11 6
11 6
12 7
12 6
11 6

-----
Process exited with return value 0
Press any key to continue . . . _
```

Mathematical Library Functions

All the arithmetical function from `#include<math.h>` header file are listed in the following table:

Function name	Operation
<code>sqrt()</code>	Return the square root of the argument
<code>pow(x,y)</code>	Return x raised to the power of y
<code>pow10(y)</code>	Return 10 raised to the power of y
<code>sin()</code>	Return the sine of argument (radian)
<code>hypot(a,b)</code>	Return the hypotenuse of a right triangle whose sides are a and b
<code>tan()</code>	Return the tangent of the argument (radians)
<code>log()</code>	Return the natural log of the argument
<code>Log10()</code>	Return the base 10 log of the argument
<code>abs()</code>	Return the absolute value of argument

Sample program using **sqrt** and **pow** function:

```
#include <iostream>
using namespace std;
#include <math.h>

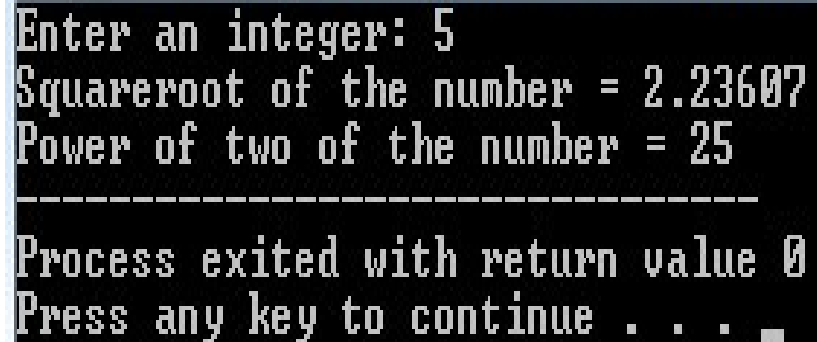
int main()
{
    int number, power;
    double squareroot;

    cout<<"Enter an integer: ";
    cin>>number;

    squareroot = sqrt(number);
    power = pow(number,2);

    cout<<"Squareroot of the number = "<<squareroot<<endl;
    cout<<"Power of two of the number = "<<power;

    return 0;
}
```



```
Enter an integer: 5
Squareroot of the number = 2.23607
Power of two of the number = 25
-----
Process exited with return value 0
Press any key to continue . . .
```

Input Statement

Data to be keyed-in through keyboard

Example:

a) *Used for numbers (int, long, float, double) or a single character (single-char)*

```
cin >> variable_name;  
cin >> value1;
```

b) *Used for strings of character (char)*

```
cin.getline(variable_name, size);  
cin>>ws;  
cin.getline(myAddress, 50);
```

c) *Used for string*

```
getline(cin, variable_name);  
cin>>ws;  
getline(cin, myAddress);
```

Input Statement (cont...)

- `cin` statement for two variables are as follows :

```
cin >> firstNumber;  
cin >> secondNumber;
```

- Or the two inputs (variables) can be combined as follows:

```
cin >> firstNumber >> secondNumber;
```


Output Statement

- `cout` statement for two variables are as follows :

```
cout << total;  
cout << finalAnswer;
```

- Or the two outputs (variables) can be combined as follows:

```
cout << total << finalAnswer;
```

Examples of Output Statement

```
//output a message (string)  
cout << "Hello world";
```

```
//output a character  
cout << 'A' ;
```

```
//output the value of a variable named average  
cout << average;
```

```
//output a number with the value 10  
cout << 10;
```

```
//output the result of addition of two integers  
cout << 2 + 5;  
or  
cout << 5 << "+" << 6 << "="<<5+6;
```

Formatting Output

- C++ allow output formatting using special commands called **escape sequences** within the **cout statement ONLY**.

Escape Sequences Character:

- Predefined symbols can be used to display better output.

Example:

<code>\n</code>	Newline.
<code>\t</code>	Horizontal tab.
<code>\r</code>	Carriage return.
<code>\a</code>	Alert sound (bell).
<code>\\</code>	Outputs a backslash character.
<code>\"</code>	Outputs a double quote character.

- **endl** means “end-of-line” is used to move the cursor to the next line right after that word.

Formatting Output (cont...)

I/O Manipulator

- Don't forget to declare the header file `#include<iomanip>`
- This header file allows you to manipulate the output stream to obtain desired effect.
- Most system divides a page of text output into twenty five rows and eighty columns.

A few common list of I/O Manipulator:

Manipulator	Action
<code>setw(n)</code>	Set field width to n
<code>setprecision(n)</code>	Sets floating-point precision to n
<code>setfill(n)</code>	Sets fill character to n

Formatting Output (cont...)

Example of using setw (n):

Create three column of output: student's name, matric number and programme.

```
#include <iostream>
using namespace std;
#include <iomanip>
#include <string.h>

int main()
{
    cout<<"\n"          // skip a line
    <<setw(30)<<"STUDENTNAME"    // display heading
        <<setw(15)<<"MATRICNO"
        <<setw(10)<<"PROGRAMME"<<endl;

    cout<<setw(30)<<"-----"    // display underline
        <<setw(15)<<"-----"
        <<setw(10)<<"-----"<<endl;

    return 0;
}
```

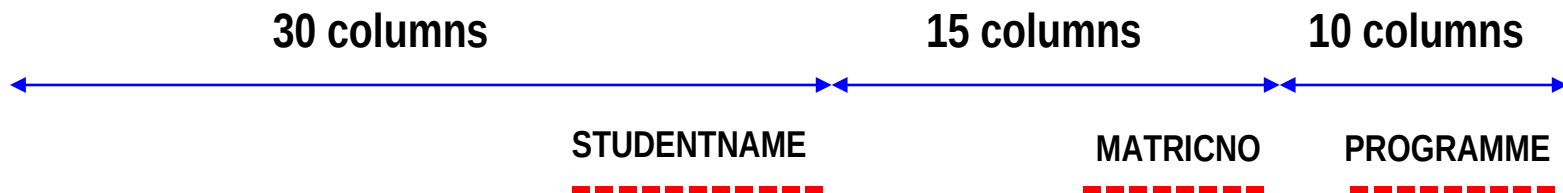
Formatting Output (cont...)

Example of using `setw (n)`:

```
cout<<"\n"           // skip a line
    <<setw(30)<<"STUDENTNAME" // display heading
    <<setw(15)<<"MATRICNO"
    <<setw(10)<<"PROGRAMME"<<endl;

cout<<setw(30)<<"-----" // display underline
    <<setw(15)<<"-----"
    <<setw(10)<<"-----"<<endl;
```

The Output Screen:



Formatting Output (cont...)

Example of using `setfill(n)`:

```
number = 101.75;  
cout << setw(10) << setfill('*') <<  
number;
```

The Output Screen:

```
****101.75
```

- Specifies character for blank space in field.
- Single quotes (`' '`) required around character enclosed in parentheses.

Formatting Output (cont...)

Generating Decimal Point Value using `setprecision(n)`:

```
cout.setf(ios::fixed);  
cout.precision(n);
```

`"cout.setf(ios::fixed)"` instruct the compiler to generate a fixed decimal output without exponential, or e, notation.

`"cout.precision(n)"` commands the **number of decimal places** to the right of decimal point to be generated.

Example:

```
int number = 101.75123;  
cout.setf(ios::fixed);  
cout.precision(2);  
cout<<number;
```

The Output Screen:

101.75

```
cout.setf(ios::fixed);  
cout.precision(n);
```


Formatting Output (cont...)

Generating Left-Justified Value

- By default, when we use `setw(n)` to print values or information, the values are right justified.
- In order to change the default, to put the outputs as left justify, we must write this statement:

```
cout.setf(ios::left) ;
```

Formatting Output (cont...)

Generating Left-Justified Value

Example Program:

```
#include<iostream>
using namespace std;
#include<iomanip>
#include<string.h>

int main()
{
    string name1="";
    string name2="";
    string prog1;
    string prog2;

    cout.setf(ios::left);

    cout<<"Enter first student name:";
    cin>>ws;
    getline(cin,name1);
    cout<<"Enter first student programme:";
    cin>>prog1;
```

Formatting Output (cont...)

Example Program (cont...):

```
    cout<<"Enter second student name:";
    cin>>ws;
    getline(cin,name2);
    cout<<"Enter second student programme:";
    cin>>prog2;

    cout<<"\n\n";
    cout<<setw(50)<<"NAME"<<setw(10)<<"PROGRAMME"<<endl;
    cout<<setw(50)<<"----"<<setw(10)<<"-----"<<endl;

    //display values
    cout<<setw(50)<<name1<<setw(10)<<prog1<<endl;
    cout<<setw(50)<<name2<<setw(10)<<prog2<<endl<<endl;

    return 0;
}
```

Formatting Output (cont...)

Generating Left-Justified Value

Output for the sample program:

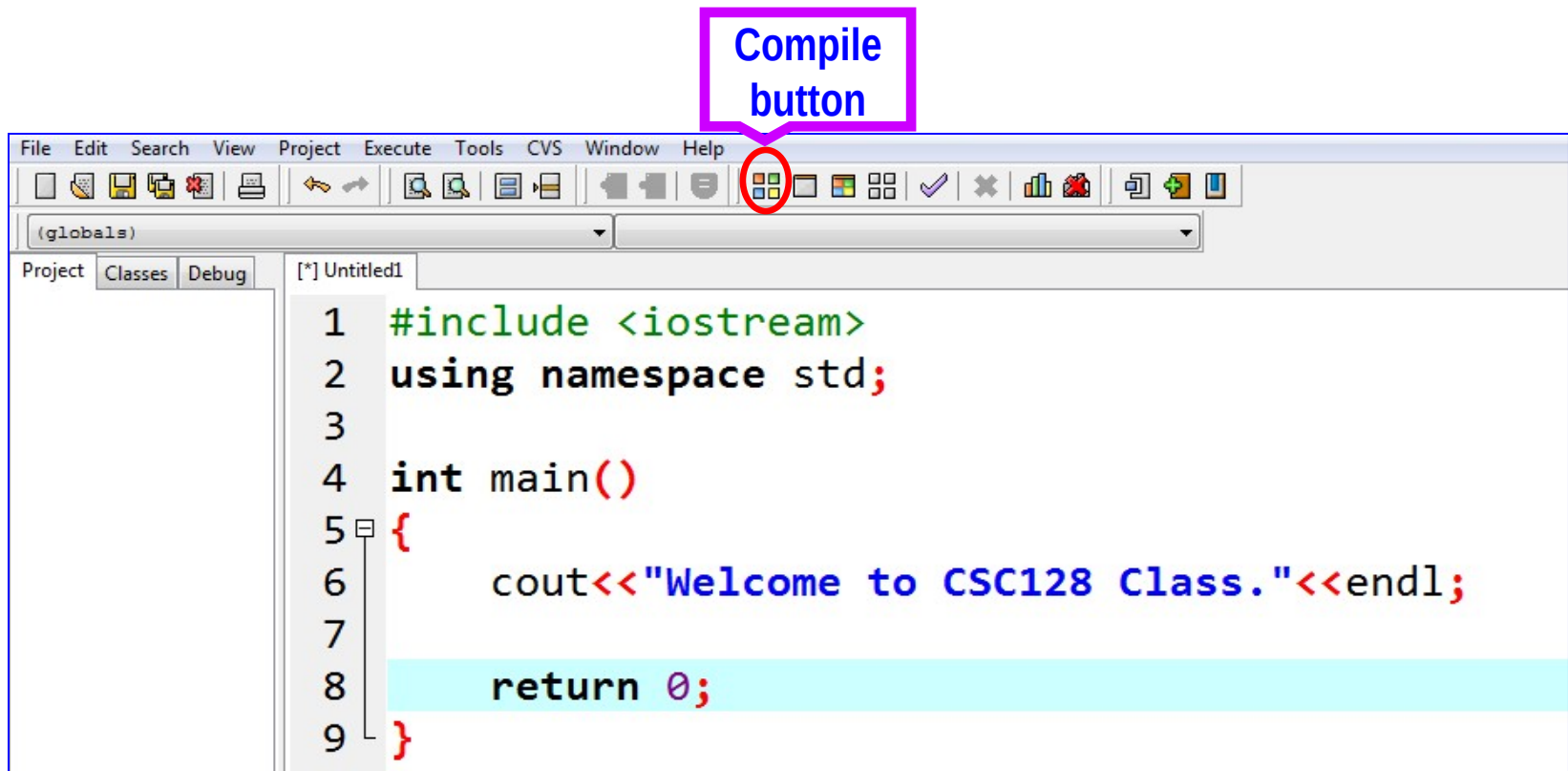
```
Enter first student name:Alya Nadiah Ismail
Enter first student programme:EM110
Enter second student name:Ahmad Hanafi Ali
Enter second student programme:EC110
```

NAME	PROGRAMME
----	-----
Alya Nadiah Ismail	EM110
Ahmad Hanafi Ali	EC110

```
-----
Process exited with return value 0
Press any key to continue . . .
```

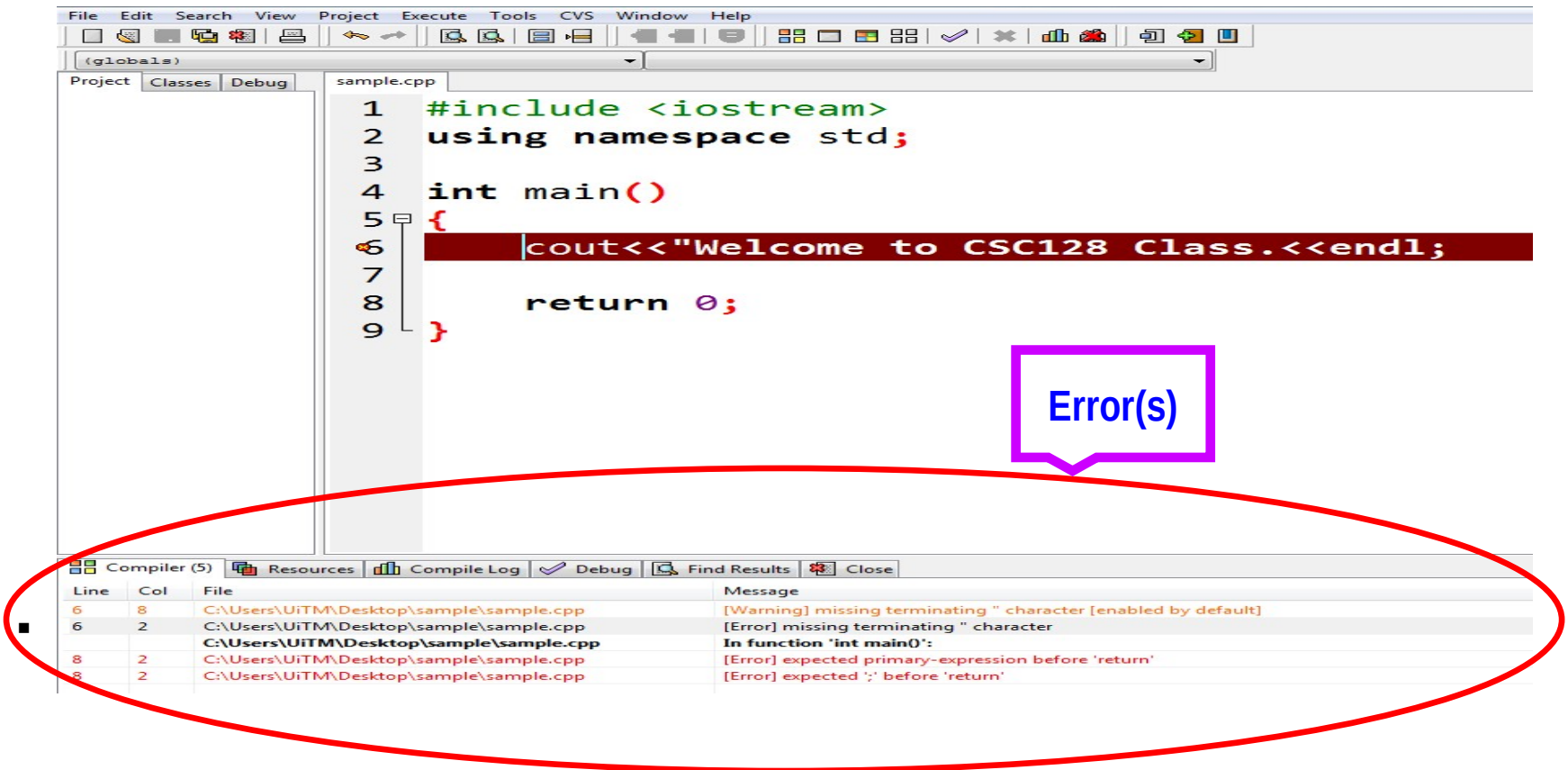
Programming Process, Debugging and Error Handling

- **Debugging** is the process of identifying and correcting errors in the programs.



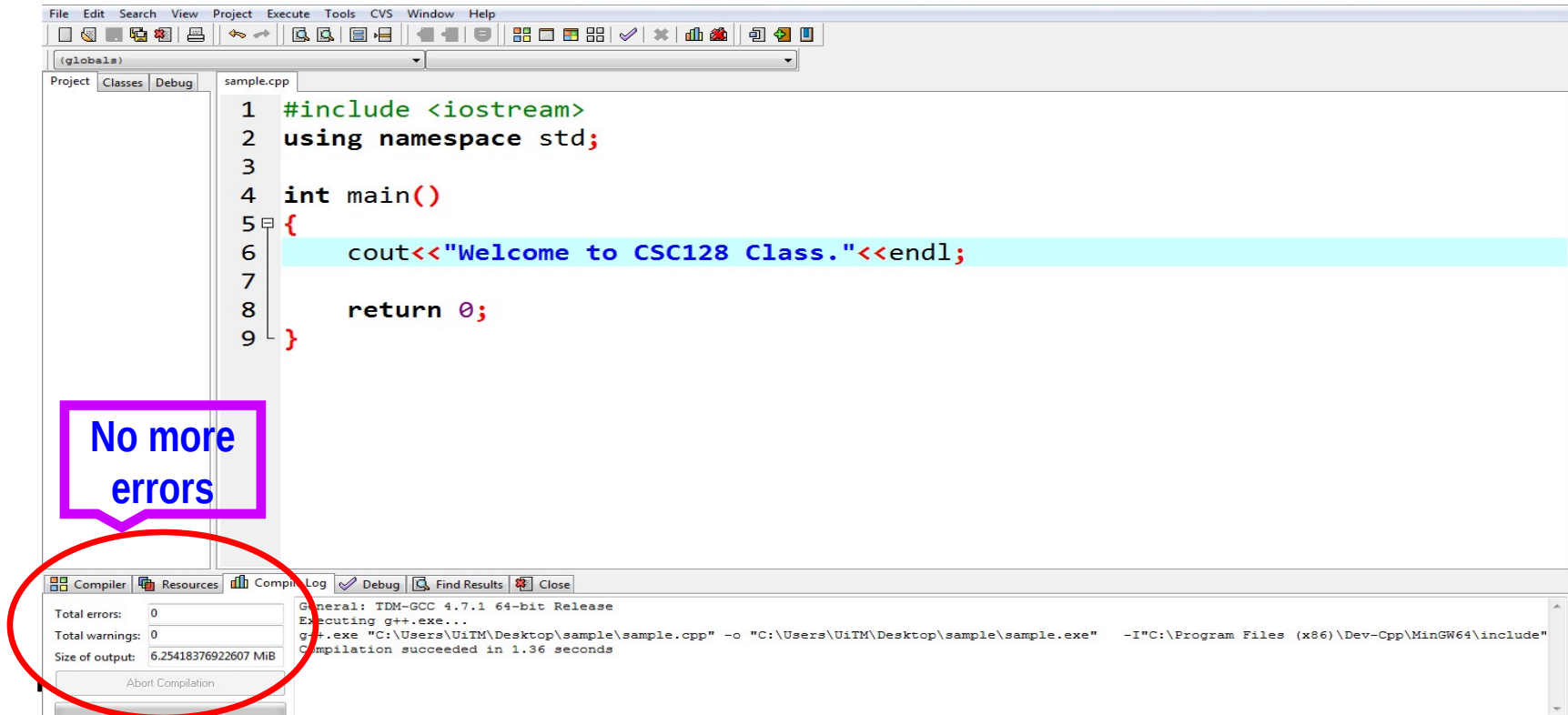
Programming Process, Debugging and Error Handling (cont...)

- Once compiled, any error (s) in the program will be displayed at the bottom of the application as below.
- The program statement which contains error will be automatically highlighted.



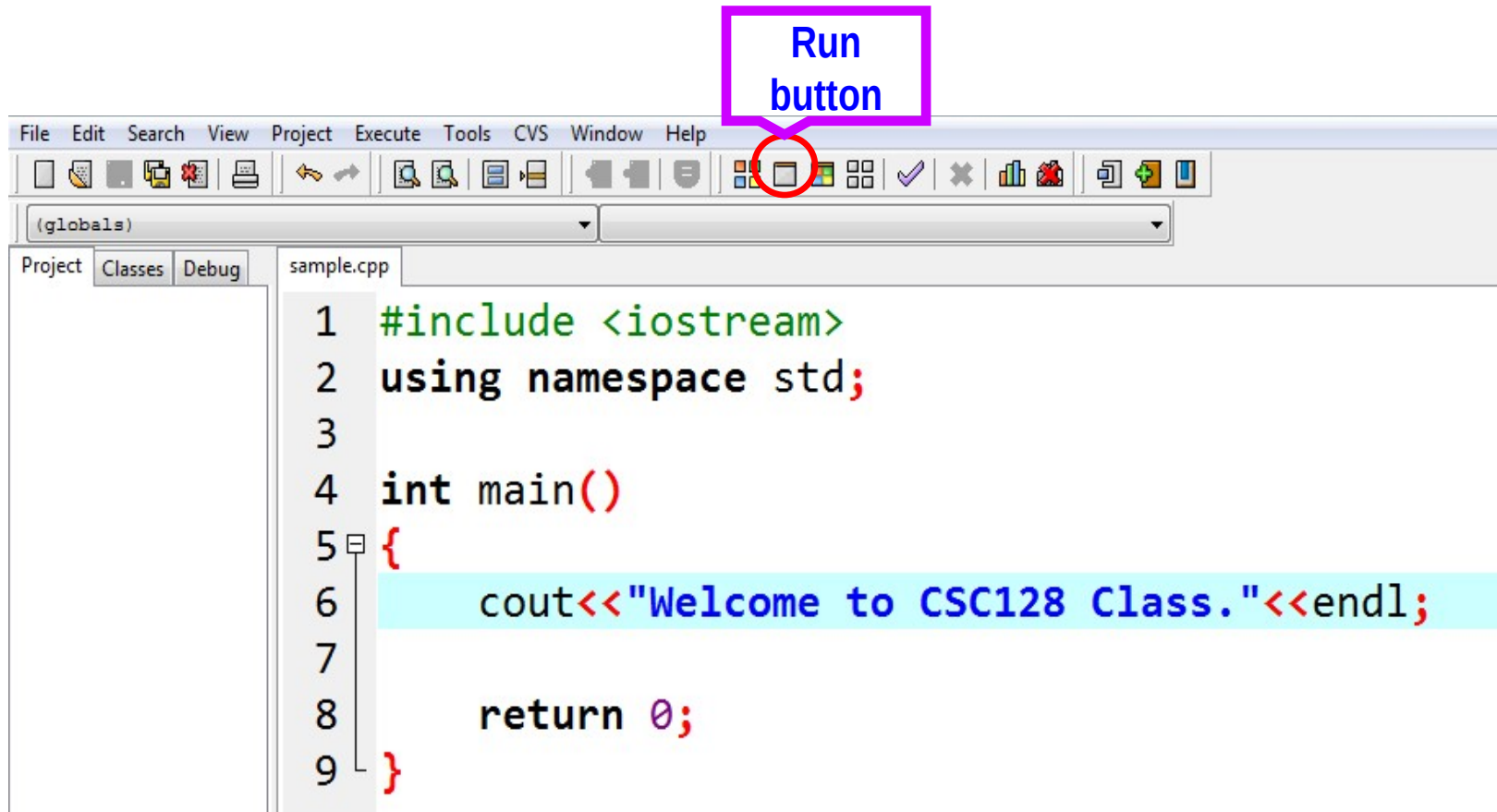
Programming Process, Debugging and Error Handling (cont...)

- After corrections are done to the program (with errors), press **compile** button again and your program is ready to be run if there are free from any errors (refer the sample below).



Programming Process, Debugging and Error Handling (cont...)

- Click run button to display your output screen.



Programming Process, Debugging and Error Handling (cont...)

▪ Types of Error

Types of Error	Description	Example
Syntax	error due to violation of programming languages.	<pre>int my age; cout>>answer; cin>>num</pre>
Logic	the real result is not the same as with the expected result or wrong output in a syntactically correct program.	
Run-time	caused program instructions that require the computer to do something illegal such as an attempt to divide a number by 0.	<pre>x=1; y=0; divide = x/y; cout<<"Division="<<divide;</pre>

Three Types Of Control Structure

- Types of Control Structures

Types of Control Structure	Description	C++ Implementation
Sequential	Statements are executed step-by-step in a linear sequence of instructions.	none
Selection / Conditional	Statements are executed based on certain condition or decision.	If If and else Switch statements
Iteration / Looping	The repetition of a statement or a group of a statements for a certain number of times or while a condition is fulfilled.	While loop Do...while loop For loop

Example of Sequential Control Structure

Question:

Calculate the subtraction and multiplication of two numbers.

Steps in problem solving:

1) To understand the following question, the first step, this question means that you have to enter any two numbers and get the subtraction and multiplication of two numbers.

2) The second step identifying the input, process and output.

Input : a, b

Process : subtract = $a - b$, assuming a always bigger than b
multiply = $a * b$

Output : subtract and multiply

Example of **Sequential** Control Structure (cont...)

- 3) The third step is drawing the flowchart or pseudo-code.
- 4) The pseudo-code is as follows :-

BEGIN

ENTER first number(a) and second number(b)

CALCULATE subtract = $a - b$

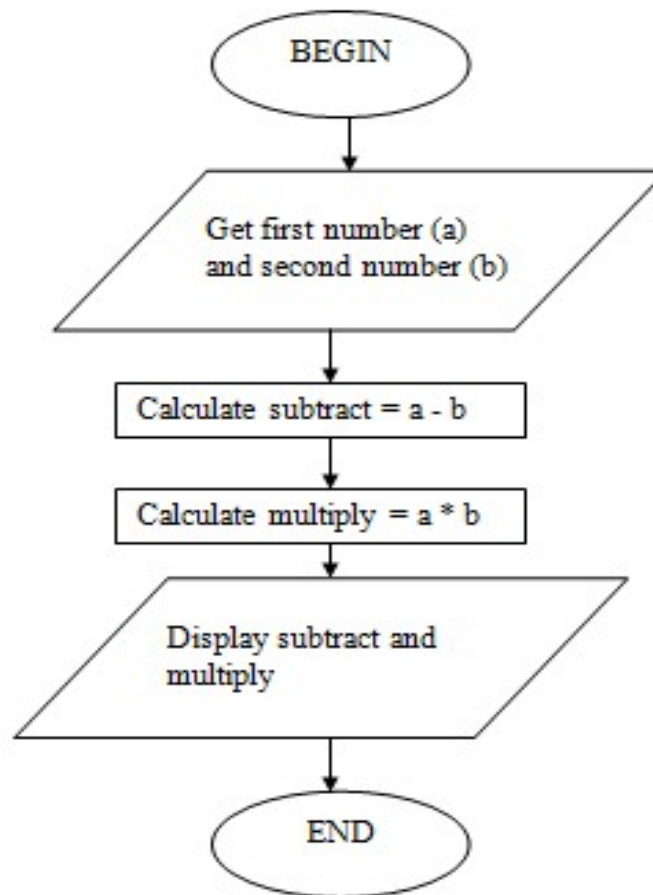
CALCULATE multiply = $a * b$

DISPLAY subtract, multiply

END

Example of Sequential Control Structure (cont...)

5) The flowchart is as follows :-



Example of Selection Control Structure

Question:

Enter the first number (a) and the second number (b). Identify which number is bigger, smaller or equal number.

Steps in problem solving:

1) To understand the following question, the first step is to understand the situation. The situation is to compare the two numbers either the number is bigger or smaller or both numbers are of equal value.

2) The second step identifying the input, process and output.

Input : a, b

Process and output : if $a > b$

Then display a bigger than b

if $b > a$

Then display b bigger than a

if $b == a$

Then display both are the same numbers.

Example of **Selection** Control Structure (cont...)

- 3) The third step is drawing the flowchart or pseudo-code.
- 4) The pseudo-code is as follows :-

BEGIN

ENTER a and b

if $a > b$

DISPLAY a is greater than b

if $b > a$

DISPLAY b is greater than a

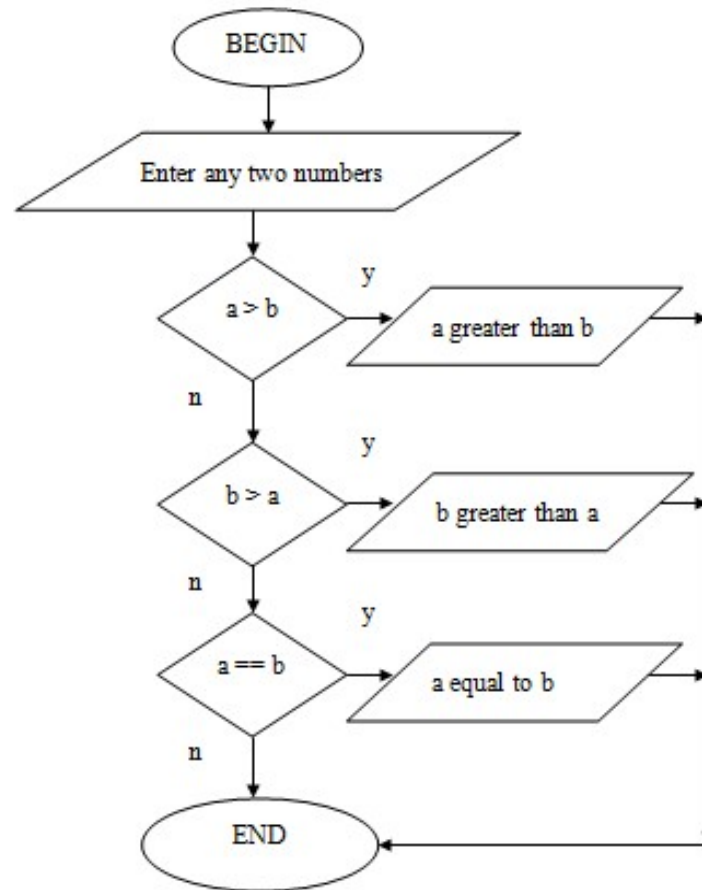
if $a == b$

DISPLAY a equal to b

END

Example of **Selection** Control Structure (cont...)

5) The flowchart is as follows :-



Example of Iteration/Loop Control Structure

Question:

Let the user key-in any numbers. User will be given a choice either to stop the program or continue key-in the next number. User must key-in 99, to stop the program. Display the number that you have enter before.

Steps in problem solving:

1)To understand the following question, the first step is to understands the situation. User will be given an opportunity to enter any numbers and total numbers are not determined. To stop the program, user must key-in the number 99.

2)The second step identifying the input, process and output.

Input : any number (a)

Process and output : while a not equal to 99

Display a

Otherwise if a == 99

Then program will stop

Example of **Iteration/Loop** Control Structure (cont...)

- 3) The third step is drawing the flowchart or pseudo-code.
- 4) The pseudo-code is as follows :-

BEGIN

ENTER any number (a)

while a not equal to 99

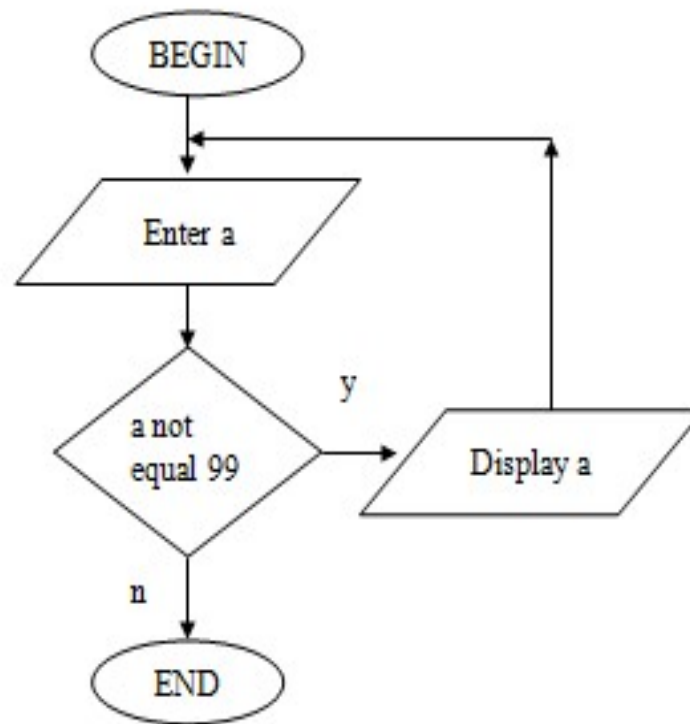
DISPLAY a

otherwise program will stop

END

Example of Iteration/Loop Control Structure (cont...)

5) The flowchart is as follows :-



Question 1

Write a C++ program to display a receipt for a restaurant. The user needs to **input cashier's name, a food menu, a food price, a drink menu and the drink price**. The program will **calculate the total price to pay** and will display the receipt as shown below:

```
C:\Users\Username\Desktop\restaurant\restaurant.exe
Enter cashier's name: Ali bin Ahmad
Enter first menu: Nasi Ayam
Price for first menu: RM 3.5
Enter second menu: Teh O Ais
Price for second menu: RM 1.50

*****
Cashier's Name :      Ali bin Ahmad
*****

-----
Menu                Price
-----
Nasi Ayam           RM 3.50
Teh O Ais           RM 1.50
-----

Dear customer, you have to pay : RM 5.00
-----
```