Principles of مبادئ علوم الحاسوب Computer

ح101، حسب 101

Introduction to C++ Programming Language

لغة البرمجة C++

محاضرة #8

Break

Quiz #1 (10 minutes

Write an algorithm that can be used to find the sum of the square of integer numbers 100,101,102,…,1000. Then draw the corresponding flowchart. (using loop)

Common Concepts

Computer Programs

It is also known as “Code”, it is a collection of instructions that can be executed to perform specific task.

Program/Code Components

Each program/code can be composed of

- **Constants**
- **Variables**
- **Procedures/Functions/methods**

Constants

Used to represented fixed values/data in the program body.

Variables

Used to represented changeable values/data in the program

Procedure/Function/Method

Can be considered as program segment that used to perform certain tasks of the whole program's tasks.

Constant/Variable Data types

Constants and variables are used in different types such as:

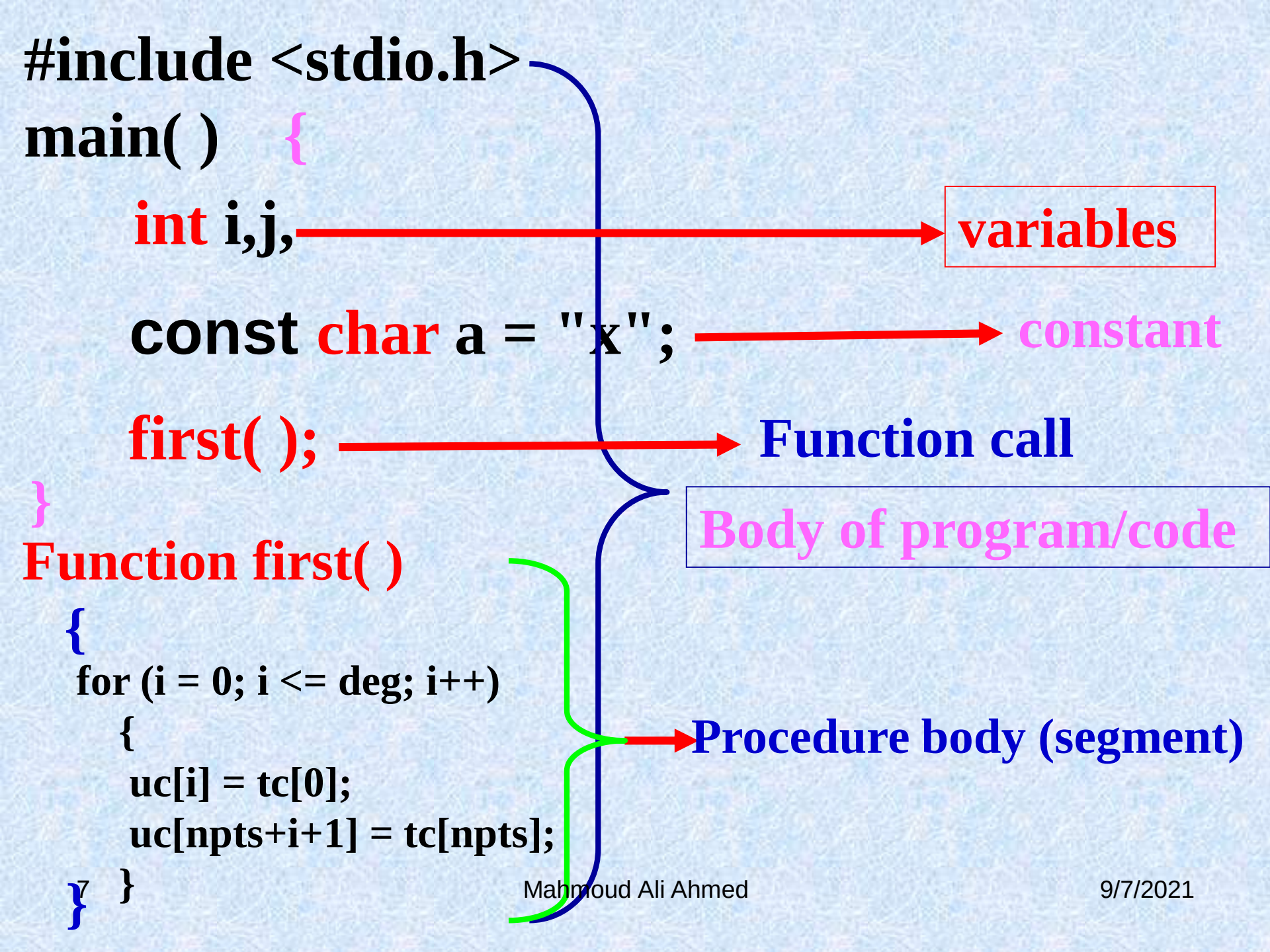
- **Character:**

The values can be composed of alphabets, numbers and other characters, such as: a, A, 0, 1, &, $, ... Also may known as text.

- **Numeric:**

The values that can be accepted by this type are all number such as: 0, 1, 2, …

Based on this, the general structure of the program can shown as:


#include <stdio.h>
main( ) {
int i,j,
variables
const char a = "x";
constant
first( );
Function call
}
Body of program/code
Function first( )
{
for (i = 0; i <= deg; i++)
{
uc[i] = tc[0];
uc[npts+i+1] = tc[npts];
} }
Procedure body (segment)

Syntax and Semantics of Programming Languages

Many of the methods and much of the terminology of linguistics apply to programming languages.

1. Syntax refers to the ways symbols may be combined to create well-formed sentences (or programs) in the language.

Syntax defines the formal relations between the elements of a language, thereby providing a structural description of the various expressions that make up legal strings in the language.

Syntax deals only with the form and structure of symbols in a language without any consideration given to their meaning.

2. Semantics reveals the meaning of syntactically valid strings in a language.

For programming languages, semantics describes the behavior that a computer follows when executing a program in the language.

Error Messages

- ***Bug*: A mistake in a program**
  - **The process of eliminating bugs is called *debugging***

***1. Syntax error*: A grammatical mistake in a program**

- **The compiler can detect these errors, and will output an error message saying what it thinks the error is, and where it thinks the error is**

Error Messages

2. ***Run-time error*:** **An error that is not detected until a program is run**

   – ***The compiler cannot detect these errors: an error message is not generated after compilation, but after execution***

3. ***Logic error*:** **A mistake in the underlying algorithm for a program**

   – ***The compiler cannot detect these errors, and no error message is generated after compilation or execution, but the program does not do what it is supposed to do***

C/C++ programming Language

Components of the Language

As any languages, C/C++ program consists of statements, any statement consists of words, any words composes of a set of letters. So what are these letters:

- **Numbers: 0, 1, …, 9**
- **Alphabet: A, B, …, Z, a, b, …, z**
- **Symbols, Special characters, and mathematical operators: ($, { . ; / + - , others**

words (names/variables) in C/C++ can be built of alphabet, numbers, and underscore (_) based on the following conditions:

- **shouldn’t be a reserved word**
- **shouldn't start with number**
- **shouldn't contain space**
- **shouldn’t contain special character rather than "_" or "$"**

Examples

- **Valid names/variables: count, a, ab1, p_1**
- **Invalid names/variables: ct*, stud name, 1a**

Note:

- **There are some words (names) not allowed to be used as user defined names. These word known reserved words such as class, main, public, void, static, and others.**
- **C/C++ is case sensitive i.e. it is differ to use capital and small letters, e.g. age & AGE.**

Primitive Data Types in C/C++

All variables and constants in any C/C++ program should follow one of the following types:

- **Character Type (char): defines variable or constant of type character, that contains one character written as: ‘A’, ‘;’, ‘1’, …**
- **Number Type: defines variable or constant of type numeric, which classified into 2 types :**

- **Integer Type:**

defines variable or constant of with integer number such as 1, 30, 100, and so on. It is classified into:

Type	Length in Memory
short	2 bytes
int	4 bytes
Long	8 bytes

- **Real Type:**

defines variable or constant of real numbers such as 1.0, 30.25, 100.75, and so on. It is classified into 2 types :

Type	Length in Memory
float	4 byte
double	8 byte

Declaring variables

Variables in C/C++ program can be declared (defined) as:

- **Initially, write down the data type**
- **Then, write down valid variable name**

Examples

char a, int a, short a, long a, float a, double a

Operators

1. Mathematical Operators

Mathematical operators performed a set of operations based on one, two, or more operands, at the end result will be returned.

Operations	Operator	example
Addition	+	a + b
Subtraction	-	a - b
Multiplication	*	a * b
Division	/	a / b
Modulus/remainder	%	a % b

2. Logical Operations

Logical operations performed through a set of operators (special symbols) on one or two operands, at the end Boolean value (true/false) will be returned.

Operation	Operator	Example
Greater than	>	x > y
Greater than or equal	>=	x>=y
Less than	<	x<y
Less than or equal	<=	x<=y
Equal	==	x==y
Not equal	!=	x!=y

Operation	Operator	Example
Logical AND	&	x & y
Logical OR	||	x || y
Logical NOT	!	!x

3. Assignment (compound) Operators

Assignment operators (special symbols) used to assign operands, a certain values.

Operation	Operator	Example
assign	=	x = y
Add then assign	+=	x+=y
Subtract then assign	-=	x-=y
Divide then assign	/=	x /= y
Multiply then assign	*=	x*=y
Modulus then assign	%=	x%=y

4.Unary Operators

Unary operators (special symbols) used to perform various operations on only one operand.

Operation	Operator	Example
Increment	++	i++
Decrement	--	i--

Documentation

Document your code using comments to make it readable and understandable.

- **Block comments**

```
/*
 * Here is a block comment.
 */
```

- **Single-Line comments**

```
/* Here is a single-line comment. */
```

- **End-of-Line comments**

```
statement; // An end-of-line comment.
```

Structure of a C++ program

Probably the best way to start learning a programming language is by writing a program. Therefore, the general structure of the program may look as:

```
// my first program in C++
# include <iostream.h>
   int main ( )  {
          statement(s);
          cout <<"Hello World !";
         return 0;
 }
```

//my first program in C ++

This is a comment line. All lines beginning with two slash signs (//) are considered comments and do not have any effect on the behavior of the program.

The programmer can use them to include short explanations or observations within the source code itself.

#include <iostream.h>

Lines beginning with a hash sign (#) are directives for the preprocessor. They are not regular code lines with expressions but indications for the compiler's preprocessor.

In this case the directive #include <iostream> tells the preprocessor to include the iostream standard file.

This specific file (iostream) includes the declarations of the basic standard input-output library in C++, and it is included because its functionality is going to be used later in the program .

int main ( )

This line corresponds to the beginning of the definition of the main function.

The main function is the point by where all C++ programs start their execution, independently of its location within the source code.

It does not matter whether there are other functions with other names defined before or after it

The instructions contained within this function's definition will always be the first ones to be executed in any C++ program.

Therefore, it is essential that all C++ programs should have a *main( )* function.

The word main is followed in the code by a pair of parentheses (( )).

That is because it is a function declaration: In C++, what differentiates a function declaration from other types of expressions are these parentheses that follow its name.

Optionally, these parentheses may enclose a list of parameters within them.

Right after these parentheses we can find the body of the main function enclosed in braces ({ }) .

What is contained within these braces (body) is what the function does when it is executed.

The contents are expressed in terms of statements.

Each statement should be ended with semicolon ";", such as: (cout <<"Hello World !";)

Standard Input / Output functions

C++ **used two standard input/output functions (*cin/cout*) defined in the class (header file) *iostream.h* to interact with the user.**

cin: **enables users to enter data using the keyboard.**

Syntax: cin >> variable_name

e.g. cin >> i; cin >> age;

Note: **variable must be declared before to be used**

cout: **enables data (output) to be displayed on the screen.**

Syntax: cout << variable_name

cout <<"text"<< variable_name

e.g. cout << **age;** ***cout <<"my age is:"<<*** **age;**

Example #1

Write C++ program that can be used to find out the difference between any two integer numbers.

Example #2

Write C++ program that can be used to find out the average score of students in 6 subjects.

Example #3

Write C++ program that can be used to find out the area of a rectangle.