


DEPARTMENT OF MECHANICAL ENGINEERING  
UNIVERSITY OF MORATUWA

CS2882  
OBJECT ORIENTED PROGRAMMING USING C++

Module Introduction and Control Structures

Dr. Manoj Ranaweera  
Senior Lecturer  
Department of Mechanical Engineering



## LEARNING OBJECTIVES

To provide the essential foundation needs for efficient design, installation and operation of IT based applications in Engineering through an understanding of the basics of Object Oriented Programming

2

## LEARNING OUTCOMES

At the end of the course the student should be able to:

- Develop programs for engineering applications in the areas of CAD/CAM, FEA, CAE, etc.
- Demonstrate programming techniques used in Object Oriented Programming.
- Demonstrate advanced skills in Object Oriented Programming required for advanced applications in IT.

3

## COURSE OUTLINE

- Introduction, Control Structures
- Arrays and functions
- Pointers and strings
- Structures and classes
- Object oriented programming
- Operator overloading and data type conversions
- Inheritance
- Console I/O operations and file handling
- Exception handling
- Introduction to UML and visual programming

4

## PURPOSE

- Why you learn Object Oriented Programming?
- How does programming help your career as a Mechanical Engineer?

5

## ASSESSMENT CRITERIA

- Final Grading:
  - Continuous Assessments : 30 %
  - End Semester Exam : 70%
- Continuous Assessments
  - Mid-term test : 10 Marks
  - Practical Test : 20 Marks

6

## NORMS AND RULES

- Punctuality is Strictly Maintained
  - Lecture : No later arrival than 4:25 pm (Thursday)
  - Laboratory: No later arrival than 10:25 am (Friday)
  - Tutorial : No Restrictions (Tuesdays)
- ADD / DROP
  - Requests for Add / Drop are not entertained after 4:00 pm on 31<sup>st</sup> of January.

7

## INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

- Microsoft Visual Studio (Version 2008 / 2010) is used as the IDE
- Create a project for EACH program you write
  - A project consists of one or many files
  - You will initially create single-file projects and then advance yourself into multi-file project of real-world type
- Let's learn how to create a project in Visual Studio 2010

8

## INTRODUCTION

- Machine language
 

```
+1300042774
+1400593419
+1200274027
```
- Assembly language
 

```
LOAD      BASEPAY
ADD        OVERPAY
STORE     GROSSPAY
```
- High-level language
 

```
grosspay= basepay +overtimepay
```

9

## SAMPLE PROGRAM

```
// A simple c program- Program 1
#include <iostream.h>
using namespace std;

void main()
{
    cout << "Welcome to C++";
    Welcome to C++
}
```

10

## C++ PROGRAMMING INTRODUCTION

- Header file?
 

```
iostream.h
conio.h
```
- Case sensitivity?
  - Value and value are different
  - C++ is case sensitive

11

## BUILT IN DATA TYPES

Data Type	Basic Data Type
int	integer
short int	integer
long int	integer
unsigned int (unsigned)	integer
unsigned short int (unsigned short)	integer
unsigned long int (unsigned long)	integer
float	Floating point number
double	Floating point number
char	character

12

## DEFINING VARIABLES

- Format  
`data_type variable_name ;`

Example:

```
int value_1;           // define an integer type variable
int value_2 = 50 ;     // define and initialize an integer variable
```

13

## C++ OPERATORS

- Arithmetic operators
- Logical operators
- Relational operators
- Incrementing / decrementing operators
- Conditional operators

14

## ARITHMETIC AND LOGICAL OPERATORS

Operation	Operator
Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus	%
Assignment	=
Equal	==

15

## LOGICAL OPERATORS

Operation	Operator
Logical AND	&&
Logical OR	
NOT	!

16

## RELATIONAL OPERATORS

Operation	Expression
Greater than	>
Less than	<
Equal	==
Not equal	!=
Greater than or equal	>=
Less than or equal	<=

17

## INCREMENTING/ DECREMENTING OPERATORS

Operator	Example	What Happens
+=	c+=3	c=c+3
-=	c-=6	c=c-6
*=	c*=4	c=c*4
/=	c/=4	c=c/3
%=	c%=7	c=c%7
++ (before)	++a	a=a+1 ;use a
-- (before)	--b	b=b-1 ;use b
++ (after)	a++	Use a ; a=a+1
-- (after)	b--	Use b ; b=b-1

18

## CONDITIONAL OPERATOR

```
if (x > 10)
    y=1;
else
    y=2;
```

➔

```
y = ( x > 10 ) ? 1:2;
```

19

## DATA TYPE CONVERSION

- Explicit conversion
  - average=`static_cast<double>`(total)/number;
  - average = `(double)`total/number;
- Implicit conversion
  - average=total/number;

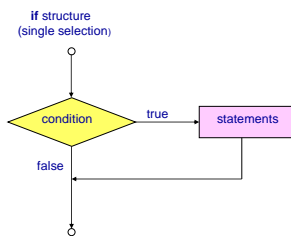
Note:

Here, *average* is a double variable while *total* and *number* are integer variables

20

## CONTROL STRUCTURES

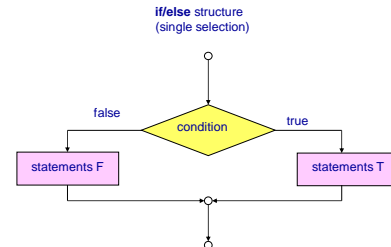
- if structure



21

## CONTROL STRUCTURES

- if/else structure



22

## CONTROL STRUCTURES

- Format of if/else

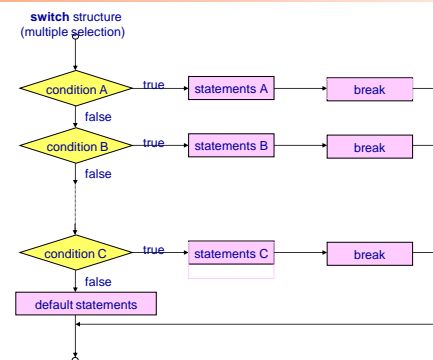
```
if(condition)
{
    //statements to be executed if the condition is true
}
else
{
    //statements to be executed if the condition is false
}
```

Note:

If there is only one statement, the bracket is optional

23

## CONTROL STRUCTURES – SWITCH/CASE



24

## CONTROL STRUCTURES

### Format of switch/case structure

```
Switch (variable)
{
case value_1:
    //statements
    break;
case value_2:
    //statements
    break;

case value_n:
    //statements
    break;
default: //optional
    //statements
    break;
}
```

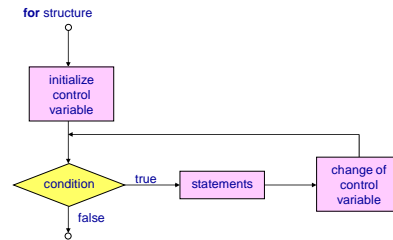
switch/case does not completely replace nested if/else



25

## CONTROL STRUCTURES

### for structure



26

## CONTROL STRUCTURES

### Format of for

```
for( initialize control variable ; condition ; change variable )
{
    //statements
}
```

Example:

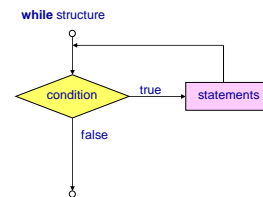
```
for(a = 0 ; a <= 5 ; a++)
{
    cout << a << "\n";
}
```

- Keyword **break** can be used at any time to leave the loop before condition expires

27

## CONTROL STRUCTURES

### while structure



28

## CONTROL STRUCTURES

### Format of while

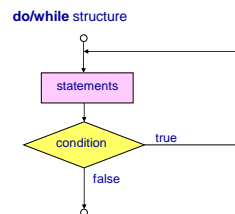
```
while( condition )
{
    //statements
}
```

- Changing of the control variable must be done within the loop
- Body is executed only if the condition is true

29

## CONTROL STRUCTURES

### do/while structure



30

## CONTROL STRUCTURES

- Format of `do/while`

```
do
{
    //statements
} while(condition);
```
- Changing of the control variable must be done within the loop
- Body is executed once before testing the condition

31

## WRAP UP

By now, you should be able to:

- Identify the essential components of a simple C++ program
  - How to include header files
  - How to structure the main function
- Use control structures
  - `for`, `while`, `do/while`, `switch/case`, `if/else`
- Get inputs from user, process them and display the results
  - `cin>>` and `cout<<`

32

## COMING UP NEXT

- Tutorial Class
  - Development of programs using the concepts learnt to solve fairly complicated real world tasks
- Practical Class
  - Practice the fundamental concepts learnt
  - Application development

33

# END

34