Advanced Programming
(BETC 1353)

Week 1 : Introduction to C++

Rosziana Binti Hashim
rosziana@utem.edu.my

Learning Outcome

At the end of this lecture, you should be able to:

- Use the basic Input / Output
- Declare and use C++ Data types
- Write general form of C++ program
- Use Arithmetic operation in C++ program
- Identify common programming errors

The History

C Language

Developed by Dennis Ritchie at AT&T Bell Labs in 1970s
- Used to maintain UNIX systems
- Many commercial applications written in C

C++ language

Developed by Bjarne Stroustrup ar AT&T Bells Labs in 1980s
- overcome several shortcomings of C
- Incorporated object oriented programming
- C++ is an “extension” of
- C language

Remain as subset of C++

Differences Between C and C++..

C++

- All function must be prototyped
- Support “Object Oriented Programming”

C

- C prototyped are recommended but technically optional

General Form of C++ program

- C++ program starts as below

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

int main()
{
    /* program code */

    return 0;
}
```

Explanation

#include <iostream>

- ***#include*** directive(Prepocessor directive) tells the compiler to include some already existing C++ code in the program
- Must include the file name (header file eg: iostream)
- The ***preprocessor*** inserts all contents of the indicated file into the program
- The include file is then linked with the program
- There are two form of #include statement ;

```
#include <iostream> //for pre-defined files

#include "my_lib.h"   //for user-defined files
```

Explanation

```
using namespace std;
```

- This statement is called a ***using directive***.
- The latest version of C++ standard divide names (e.g. cin and cout) into sub collections of names called namespaces.
- This particular using directive says the program will be using names that have a meaning defined for them in the ***std*** namespace
- In this case the iostream header defines meaning for cout and cin in the std namespace

Basic Input Output : cout

Use the cout object to display information on the computer's screen

```
cout << expression;
```

Consider the following program statement

```
cout <<"Hello World"<< endl;
```

- **cout** (see-out) used for output to the monitor
  - it is a stream object

- "<<" operator is used to send string "HELLO World " to **cout**.
- In the case the << symbol is called the stream insertion operator

Stream manipulator **endl** ("end-line") cause a new line to be started on the monitor

Basic Input Output : cin

The cin object is useful for reading data from the keyboard

```
cin >> variables;
```

Consider the following program statement

```
cin >> number_of_pods;
```

- **cin** (see-in) used for input from the keyboard
- it get characters from the stream object on the left of ">>" and then stores it to the variable

- ">>" is the stream extraction operator
- The cin objects will continue waiting new data until the [ENTER] key is pressed

Basic Input Output : cin

cin object : Deal with **`string`** and **`char`**

- The problem while using `cin` with the >> operator for entering **strings**

  - Any leading whitespace characters (space, tabs or line break will be ignores and passes over)
  - It stop reading when it comes to the next whitespace character

USE

getline (cin, inputLine)

- `getline` function :

  Reads an entire line, including any white spaces and stores it in `string` objects .

- `cin` :

  The input stream we are reading from

- `inputLine` :

  The name of the `string` object receiving the input

Basic Input Output : cin

cin object : Deal with **`string`** and **`char`**

- The problem while use `cin` with the >> operator for entering **strings**

  - It is impossible to input just a whitespace or [ENTER] with cin >> since it passes over all whitespaces
  - The cin statement will not be passed until some characters others than [ENTER] key, tab key or spacebar has been pressed.

USE

```
cin.get(ch) ; or
ch = cin.get() ;
```

- `get` member function :

  Reads any single character.

- `Ch` :

  The name of a char variable that the character is being read into

Basic Input Output : cin

Mixing `cin` >> and `cin.get`

- A problem will occurred when mixing `cin` >> with `cin.get` because `cin` >> and `cin.get` use slightly different techniques for reading data.

```
cin.ignore(n, c) ;
```

- `cin.ignore` function tell the cin object to ignore one or more characters entered using the keyboard.
- n is an integer and c is a character.

Example:

```
cin.ignore(10, '\n') ;
```

cin will ignore the next 10 characters or until a newline is encountered, whichever come first

- If no arguments are used, `cin` will skip only the very next character :

```
cin.ignore () ;
```

Basic Command of Output

Escape Sequence	Name	Description
`\t`	Horizontal Tab	Takes the cursor to the next tab stop
`\a`	Audible alert	sound
`\\`	Backslash	Displays a backslash (\)
`\"`	Double Quote	Displays a quotation mark (")
`\v`	Vertical Tab	Takes the cursor to the next tab stop vertically.
`\'`	Apostrophe	Displays an apostrophe (')
`\n` or `endl`	New line	Takes the cursor to the beginning of the next line
`\?`	Question mark	Displays a question mark

C++ Identifiers

- An **identifier** is a name for
  - **variable**
  - **constant**
  - **function**, etc.
- It consists of a **letter** followed by any
  - **sequence of letters**
  - **digits**
  - **underscores**.
- Examples of valid identifiers: *father_name, year, y2015*
- Examples of invalid identifiers: *2015y*
- special characters are prohibited.

  example: *a=c, J-25, &Ricky,*Jackson*

C++ Identifiers

- It allows programmers to name data and other objects in the program such as constant, function, variable etc.
- Any capital letter (A~Z), lowercase letters (a~z), digits (0~9) and also underscore (_) can be used

- Identifier's Rules
  - The first character must be alphabetic character or underscore
  - Only alphabetic characters, digits and underscores are allowed and cannot contain spaces
  - Duplicate any reserved word is prohibited

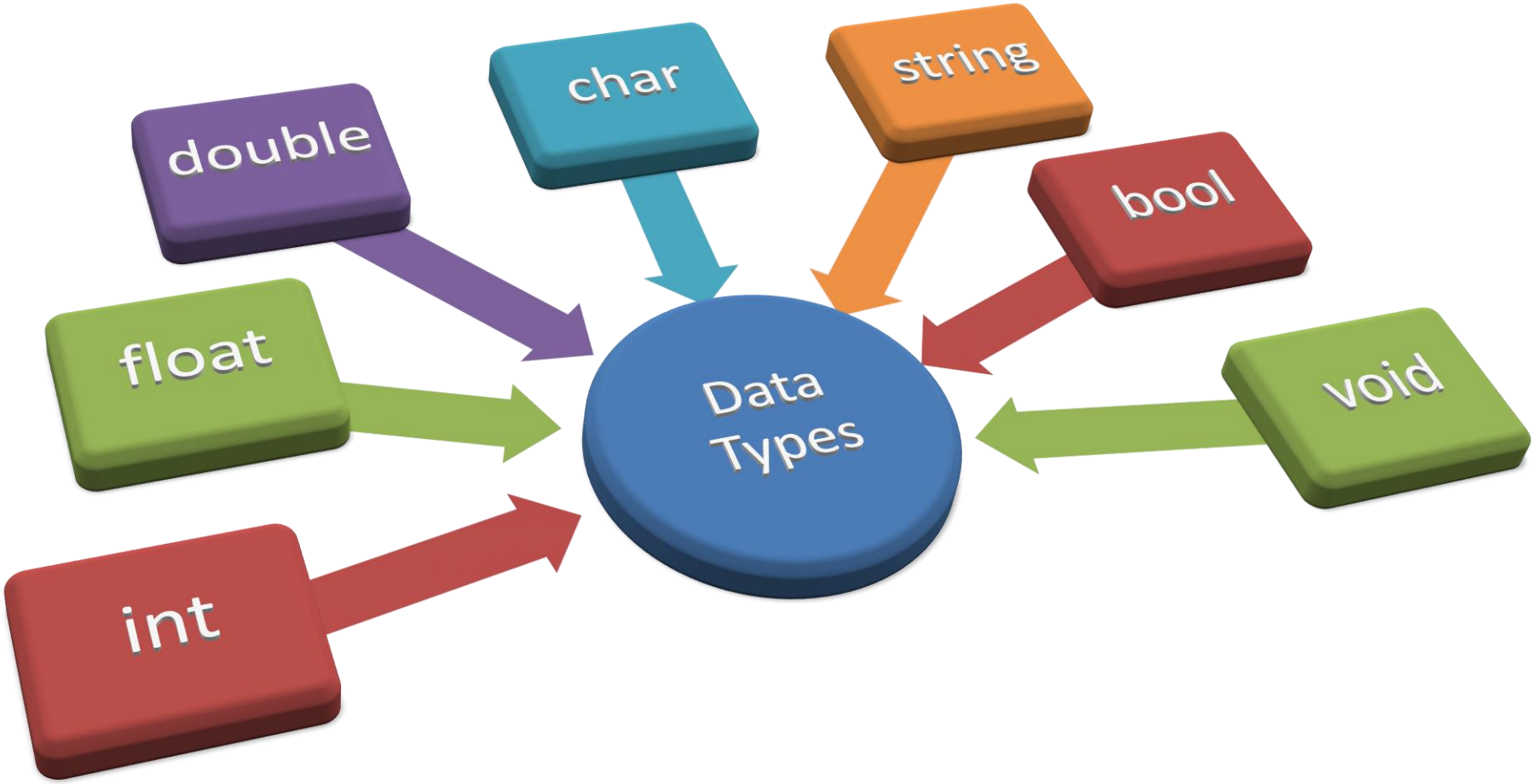
- C++ is case-sensitive; this means that CASE, Case, case, and CaSe are four completely different words.

C++ Keywords

- Each keyword has a predefined purpose in the language.
- All reserves word appear in lowercase.
- Keywords is prohibited to be used as variable and constant names!!
- Some of the keyword that we will cover this class:

```
bool, break, case, char, const,
continue, do, default, double, else,
extern, false, float, for, if, int,
long, namespace, return, short,
static, struct, switch, typedef,
true, unsigned, void, while
```

C++ Datatypes



int

- Stands for integer
- Storing positive or negative whole number
  - Fractional part is prohibited
- Examples:

```
int integerNumber; //declaring an integer

integerNumber = -123; //assign to negative
integerNumber = 547;  //assign to positive
```

float

- Stands for floating point number
  - Allowing fractional part
  - About 7 digits of precision
- Can store negative or postive number
- Examples:

```
float floatNumber;  // declare a float

floatNumber = -4.95412; //assign negative value
floatNumber = 11.91253; //assign positive value
```

double

- Similar to float, but have twice the digits of precision
- Examples:

```
double doubleNum;  // declare a double

doubleNum = -4.954119654123;
doubleNum = 11.912533651204;
```

char

- Variable to handle single character:
  - Letters
  - Digits
  - Special characters
- Basically, any character listed in ASCII table
- Enclosed by single quotes ‘ ’.
- Example:

```
char myChar;  // declare a char

myChar = 'k'; // assign to a letter
myChar = '0'; // assign to a digit
```

string

- Basically, an array of multiple characters
- Enclosed by double quotes “ ”
- Example:

```
string firstString;
string lastString;

firstString = “Machu”;
lastString = “Picchu”;

cout << firstString + lastString ;
```

bool

- Boolean holding only two values: TRUE or FALSE
  - Literally an integer with value 1 or 0, respectively
- Usually used in condition statements (if-else)
- Examples:

```
bool carLock;  // declare a boolean

carLock = FALSE; // car is unlocked
carLock = TRUE;  // car is locked
```

void

- For identifiers which has no values
- Used by function which does not returning any output
- Examples:

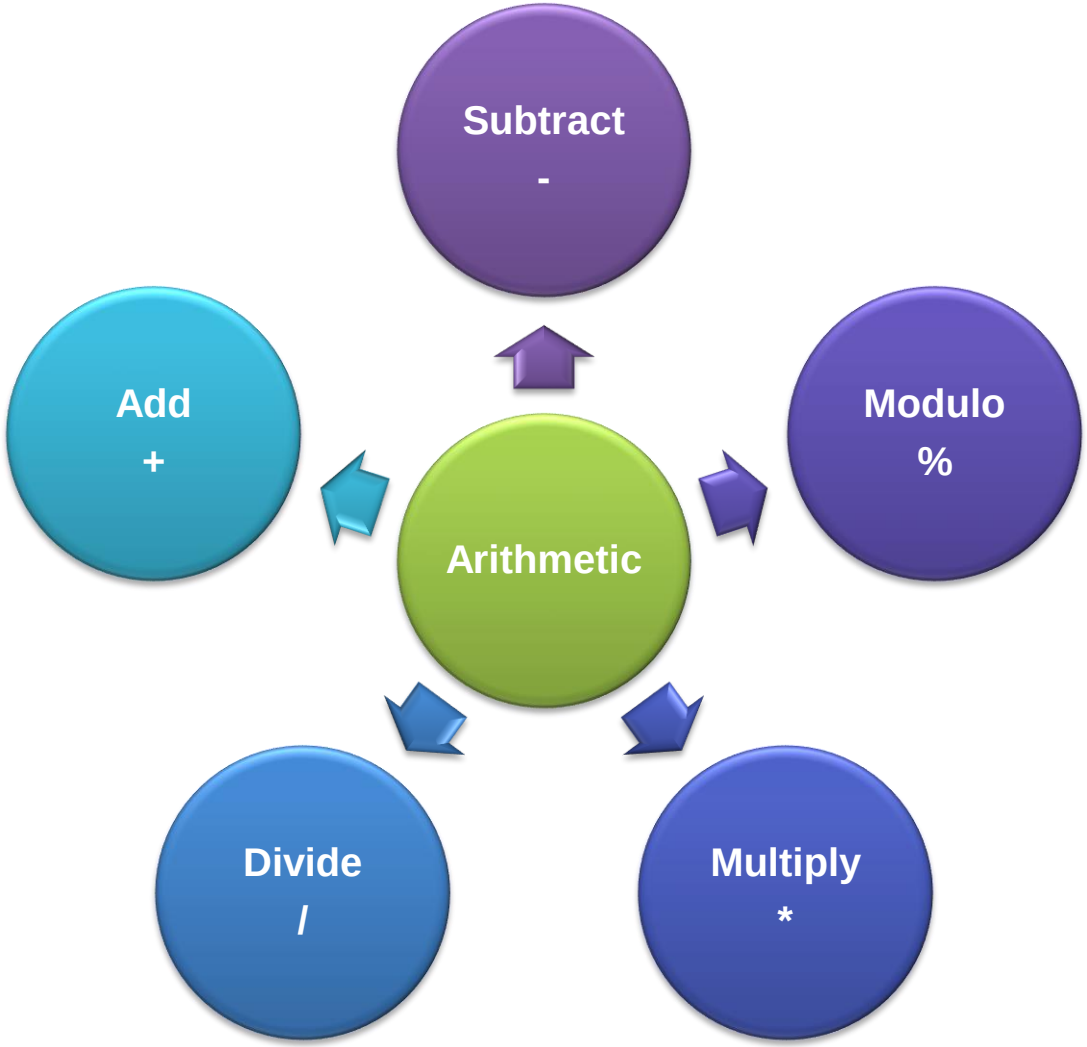
```
void DisplayHelloMessage()
{
   cout << " Hello everyone!!" << endl;
   cout << " Welcome to my program!" << endl;
}
```

Summary on C++ Datatype

Name	Description	Size in byte
int	Integer number, positive or negative	4
char	Single character	1
float	Floating number	4
double	Double-precision floating number	8
bool	Boolean value (FALSE or TRUE)	1

operators

Arithmetic operators



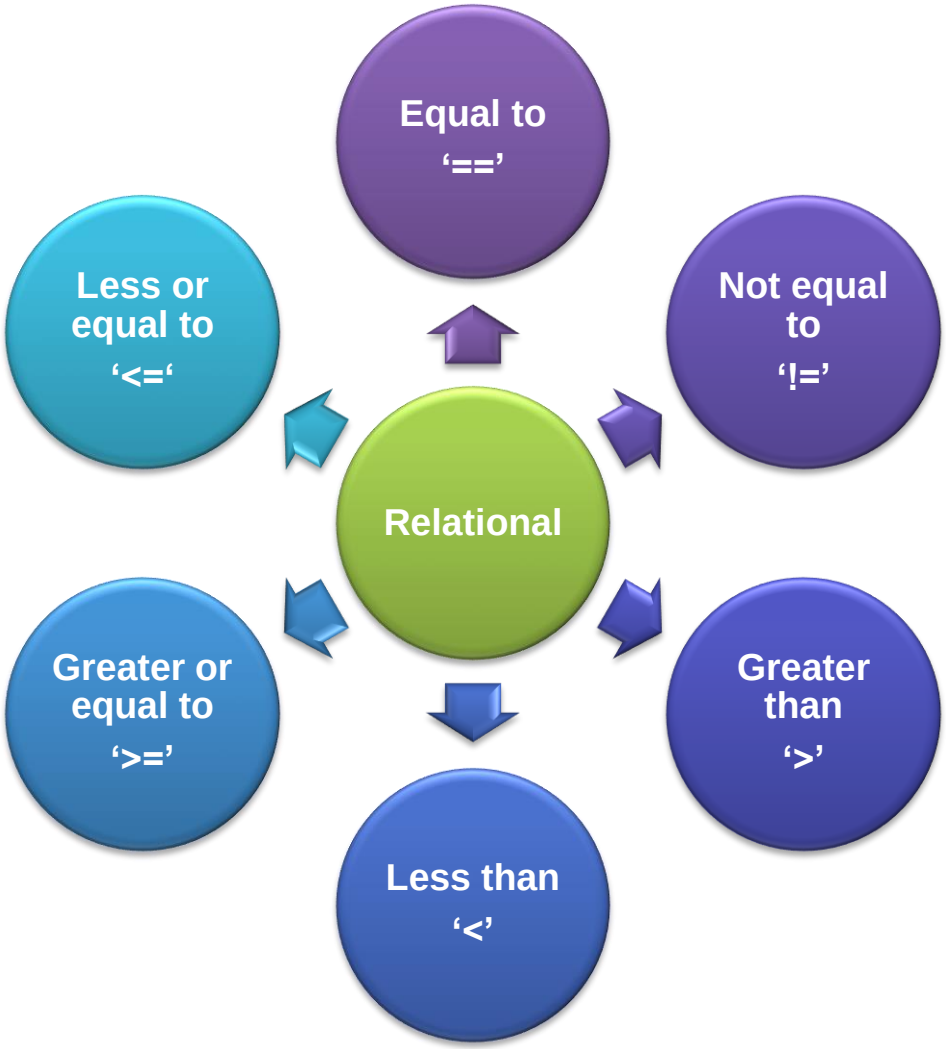
Arithmetic operators

- Example:

```
int var_1 = 3;
int var_2 = 6;
int res_1;
int res_2;

res_1 = var_2 + var_1; //res_1 = 6+3 = 9
res_2 = var_1 * var_2; //res_2 = 3*6 = 18
```

Relational operators



Relational operators

- Example:

```
cout << "Enter your age: ";
cin >> myAge;

if (myAge > 60)
{
  cout << "You must retire already!";
}
else
{
  cout << "You're still fit."
}
```

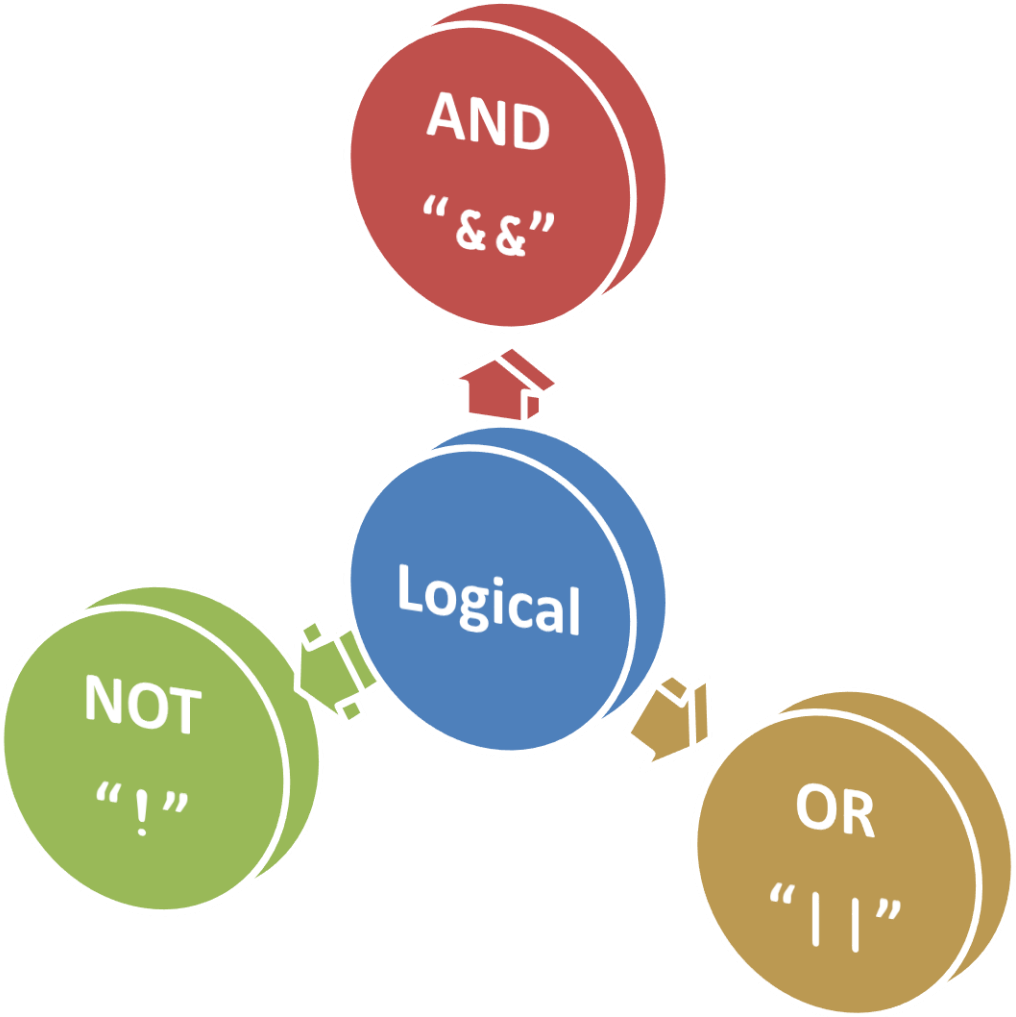
Relational operators

Output:

```
Enter your age: 65
You must retire already!
```

```
Enter your age: 35
You're still fit.
```

Logical operators



Logical operators

- **Example:**

```
cout << "How old are you? ";
cin >> myAge;
cout << "What is your gender? ";
cin >> myGender;

if (myAge < 30 && myGender == 'm'){
   cout << "You are handsome and young!";
} else if (myAge < 30 && myGender == 'f'){
   cout << "You must be a good looking young girl!"
} else {
   cout << "You can take care of your appearance."
}
```

Logical operators

Output:

```
How old are you? 26
What is your gender? m
You are handsome and young!”
```

```
How old are you? 35
What is your gender? f
You can take care of your appearance.
```

C++ comments

- explanatory notes
- ignored by the compiler.
- Two methods of commenting within a program:
  - **Using double slash //**

    Ex: `// Explain on the following statements`

  - **Using /* */**

    Ex: `/* Commenting a portion of the code. This is suitable for commenting multiple lines within the program.*/`

Programming Errors

- Syntax errors
  - Violation of programming language grammar rules
  - Detected and notified by the compiler
  - Examples:
    - Missing semicolon ';' , undeclared identifiers, etc.
- Run-time errors
  - Errors detected by the system during running time
  - Examples:
    - Insufficient memory
    - Segmentation fault
- Logical errors
  - Mistakes in the implemented algorithm
  - Undetectable by the compiler or computers, only known to the programmers

Self-Review Questions

Question 1

Determine whether the following statements are valid or not?

a) int x = ‘3’;
b) float m = 3.12454;
c) char n = 1;
d) string mystr = “hello friends”;

Self-Review Questions

Answer:

a) `int x = '3';`

ans: invalid since a character is assigned to an int variable

b) `float m = 3.12454;`

ans: valid

c) `char n = 1;`

ans: invalid since an integer is assigned to a char variable

d) `string mystr = "hello friends";`

ans: valid

Self-Review Questions

Question 2

Write a program to calculate and the display the area of a rectangle. The length and width must be entered from the keyboard.

Self-Review Questions

Answer

```
int area, length, width;
cout << "Please enter the length: " ;
cin >> length;
cout << "\nPlease enter the width: ";
cin >> width;

area = width * length;
cout << "\n Area of rectangle is " << area << endl;
```