CSE 5910
C++ Basics

Instructor: Aijun An
Department of Computer Science and Engineering
York University
aan@cse.yorku.ca

http://www.cse.yorku.ca/course/5910

1

Outline

Variables
Data types
Constants
Expressions
Assignment statements
I/O statements
Lab exercises

2

C++ Program Structure

A C++ program consists of
functions, variables, constants, classes,…

A function contains
- *variable declarations*
- *statements* that operate upon variables and constants

Every program must contain a **`main()`** function
- Execution of a program starts at **`main()`**

3

A Simple C++ Program

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

int main()
{
  int number1, number2;
  int sum;

  cout << "Please input two numbers: ";
  cin >> number1 >> number2;
  sum = number1 + number2;
  cout << "The sum of these two number is ";
  cout << sum << "\n";

  return 0;
}
```

Directives to include declarations of some variables and functions used in the program but defined elsewhere, e.g., library.

main function

Variable declarations

Statements

4

Variables

Variables store data, *whose value can change*.

Variables are like small blackboards
- We can write a number on them
- We can change the number
- We can erase the number

C++ variables are names for memory locations
- We can write a value in them
- We can change the value stored there
- We cannot erase the value in the memory location
  - Some value is always there

5

Variable Names

Each variable is identified by its name.
- In C++, names of *variables*, *functions*, *classes*, etc. are called ***identifiers***.

Rules for choosing variable names
- First character must be
  - a letter, or
  - the underscore character
- Remaining characters must be
  - letters
  - numbers
  - underscore character
- Length can be 1 or more. No length limit with **`g++`** compiler.
- Use meaningful names that represent data to be stored

6

Variable Names (*Cont'd*)

Names are *case sensitive*

- "**a**" and "**A**" are not the same

Names starting with "_" are usually reserved (library names)

Cannot use ***keywords*** as identifiers

- Keywords are reserved words used in the C++ language:
  if, for, while, float, double, int, char, long, break, true, false, const……
- They must be used as they are defined in the programming language

7

Exercise

Indicate which variable names are legal and which are not:

- **ace_5**
- **5_ace**
- **_ace5** *ok but dangerous*
- **i.g**
- **x-y**
- **X2Y**
- **int**

8

Exercise

Which of the following is a legal identifier?

- **5_And_10**
- **Five_&_Ten**
- **__________**
- **LovePotion#9**
- **"Hello World"**

9

Exercise

Can you give good variable names to store

- the speed of an automobile?
- an hourly pay rate?
- the highest score on an exam?

10

Data Types

Variables and values have types in C++

Below are some of the basic types in C++:

- **int** – integer (e.g., 3, 102, 3211, -456, etc.)
- **float** - single-precision floating point number
  - real number, i.e., number with a fractional component
  - Example: 1.34, 4.0, -345.6, 0.2453, etc.
- **double** - double-precision floating point number
  - real number.
  - Can have more significant digits than the **float** type
  - Example: 1.34, - 235.67, 21.368268683526, -0.286286382621
- **char** – character, a single byte
  - Example: '**b**', '**A**', '**#**', '**8**', '**-**'
- **bool** – Boolean type with two values: **true** or **false**

11 We will discuss other types later.

Data Types (*Cont'd*)

Different types of data

- occupy different sizes of memory space
- have different internal representations in memory.

(The values in the table are a sample. The values vary among systems.)

Type name	Memory used	Value range	Precision (# of significant digits)
char	1 byte	N/A	N/A
int	4 bytes	$-2^{31} \sim 2^{31}-1$	N/A
float	4 bytes	Approximately $-10^{38} \sim -10^{-38}$ $10^{-38} \sim 10^{38}$	7 digits
double	8 bytes	Approximately $-10^{308} \sim -10^{-308}$ $10^{-308} \sim 10^{308}$	15 digits

12

Declaring Variables

Before use, variables must be declared

- Tells the compiler the type of data to store

Declaration syntax:

- ***Type_name Variable_1, Variable_2, . . . ;***

Examples:

```
int  number_of_bars;
double one_weight, total_weight;
```

- **number_of_bars** is of type integer
- **one_weight** and **total_weight** are both of type double

13

Declaring Variables (*Cont'd*)

Locations for variable declarations

- At the beginning of a function:

```
int main()
{
  int sum;
  int score1, score2;

  cin >> score1 >> score2;
  sum = score1 + score2;
  cout << sum << "\n";
  return 0;
}
```

- Immediately before use:

```
int main()
{
  int score1, score2;
  cin >> score1 >> score2;

  int sum;
  sum = score1 + score2;
  cout << sum << "\n";
  return 0;
}
```

Both places are ok in C++. But it is a good practice to define variables at the beginning.

14

Declaring Variables (*Cont'd*)

Declaring a variable tells the compiler to locate a memory space for the variable

- The initial value of the variable depends on what's in that memory space.

You can give an initial value to a variable when declaring it. You can do it with one of two methods.

- Method 1:

```
double mpg = 26.3, area = 0.0, volume;
```

- Method 2:

```
double mpg(26.3), area(0.0), volume;
```

With either method, **mpg** is initialized to 26.3 and **area** is initialized to 0

15

Outline

Variables

Data types

Constants

Expressions

Assignment statements

I/O statements

Lab exercises

16

Constants

Constants refer to fixed values that the program may not alter.

- Examples: **3.14, -10, 'C', "hello", true**,
- They are called ***literals***.

Constants can be of any of the basic data types and can be divided into

- Integer constants
- Double constants
- Characters
- Strings
- Boolean values
  - Two values: **true** and **false**

17

Integer Constants

Integer constants do not contain decimal points

Can be written in one of the three forms

- Decimal form (base 10):
  - **2, 198, 0, -9823,** etc.
- Octal form (base 8):
  - **012, 025, -025**, etc. // start with zero
- Hexadecimal form (based 16)
  - **0x4b**, **0X7c**, etc. // start with 0x or 0X

18

Double Constants

Double constants can be written in two ways

- Simple form must include a decimal point
  - Examples: `34.1` `23.0034` `1.0` `89.9`
- E-notation (Floating point notation, Scientific notation)
  - Used to write very large or small values
  - Examples:
    - `3.67e17` means $3.67 \times 10^{17} = 367000000000000000.0$
    - `5.89e-6` means $5.89 \times 10^{-6} = 0.00000589$
    - `-30.41e2` means $-30.41 \times 10^{2} = -3041$

    Here **e** (or **E**) represents "***times ten raised to the power of***"
  - Number left of **e** does not require a decimal point
    - `3e5` is legal
  - Exponent cannot contain a decimal point
    - `3e5.1` is not legal

19

Integer and Double Constants

2 and 2.0 are not the same number in the internal representation

- 2 is of type int (4 bytes)
- 2.0 is of type double (8 bytes)

Numbers of type int are stored as exact values

Numbers of type double may be stored as approximate values due to limitations on number of significant digits that can be represented

20

Character Constants

A C++ character is one byte (8-bit) in size

A constant character is specified with single quotes:

- Regular characters

`'A', 'C', 'z', '2', '#', '$', ….`

Example:

```
char x='A', y='$';
```

- Special characters: invisible or control characters
  - Use ***escape sequence*** to represent
  - `'\n'` represents "newline", `'\t'` represents "tab", `'\0'` represent the null character.

Example:

```
char z='\t';
```

21

Special Characters

Escape sequence	Meaning
`\n`	New line
`\t`	Tab
`\0`	The null character
`\\`	The \ character
`\"`	Double quote
`\'`	Single quote

22

Internal Representation of Characters

They are integers, interpreted according to the character encoding (usually ASCII),

- e.g. `'a'` is 97,
  `'0'` is 48

```
char c;
c='0'; same as
c=48;
```

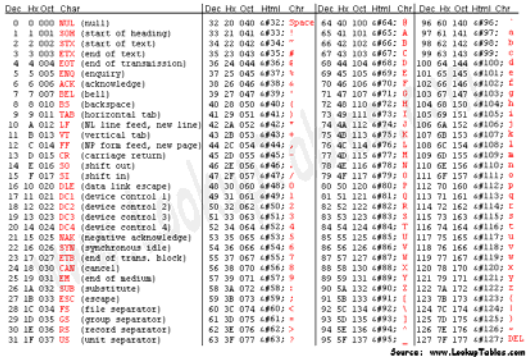
Escape sequences are integers too

- e.g. `'\n'` is 10 (newline character)
  `'\t'` is 9 (horizontal tab)

23

Internal Representation of Characters

ASCII Table



24

String Constants

A string constant is a sequence of characters in double quotes

```
"This is a string!"
```

A string can contain control characters (escape sequences)

```
"\tThis is also a string!\n"
```

In memory, a string constant is a sequence of bytes terminated by a null char `'\0'`

`"a"` --- 2 bytes

`"\tThis\n"` --- ? bytes

25

Naming Constants

Sometimes, it is better to give a constant a name to show the meaning of the constant.

In C++ we can use keyword **`const`** to *declare a constant*

Example:

```
const double PI = 3.14;
```

declares a constant named **`PI`**.

- Its value cannot be changed by the program like a variable
  - Compiler will show error if try to change its value.
- Benefits:
  - Meaningful, and it is easy to edit the value in the program if it's used multiple times.
- It is common to name constants with all capitals

26

Outline

Variables

Data types

Constants

Expressions

Assignment statements

I/O statements

Lab exercises

27

Expressions

Expressions are formed by combining variables, constants and function calls using operators.

They are used for computation.

- Each expression can be evaluated to a value.

Arithmetic operators in C++

- `+` for addition
- `-` for subtraction
- `*` for multiplication
- `/` for division
- `%` for remainder (modulo)

Examples of arithmetic expressions: *Function call*

```
score1+score2,  3*x-2,  sqrt(y)/3.5
```

28

Examples of Arithmetic Expressions

Arithmetic Expressions

Mathematical Formula	C++ Expression
$b^2 - 4ac$	`b*b - 4*a*c`
$x(y+z)$	`x*(y + z)`
$\frac{1}{x^2+x+3}$	`1/(x*x + x + 3)`
$\frac{a+b}{c-d}$	`(a + b)/(c - d)`

Sli

Introduction

Remainder Operator

% operator gives the remainder from integer division

Example:

```
5 % 3 is 2
12 % 3 is 0
14 % 3 is 2
```

Integer Division

30

Arithmetic Expressions

You can use spacing to make expressions more readable

- Which is easier to read?

`x+y*z` or `x + y * z`

Precedence rules for operators are the same as used in your algebra classes

Use parentheses to alter the order of operations

`x + y * z` (y is multiplied by z first)
`(x + y) * z` (x and y are added first)

31

Precedence

Appendix 2 in the textbook gives a full table of precedence of operators

For arithmetic operators, it is easy:

- parentheses first
- `/, *, %` before `+, -`
- `+, -` before assignment operator ('=')

```
y = 1 + 3*x;
```

32

Results of Arithmetic Operation

Arithmetic operators can be used with any *numeric type* (e.g., `int, float, double`)

An operand is a number or variable used by the operator

```
score1 + score2
```

operand *operator* *operand*

Result of an operation depends on the types of operands. For example,

- If both operands are int, the result is int
- If one or both operands are double, the result is double

33

Division of Integers

Be careful with the division operator!

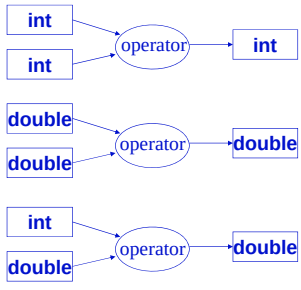
- `int/int` produces an integer result (true for variables or numeric constants)

```
int a, b, c;
a = 5;
b = 3;
c = a / b;
```

- The value of `c` is 1, not 1.666…
- Integer division does not round the result, **the fractional part is discarded!**

34

Result of Operation - Examples



17/5
- 3

17.0/5
- 3.4

9/2/3.0/4
- 9/2=4
- 4/3.0=1.33333
- 1.3333/4=0.333333

35

General Rule: Type Promotion

Given an expression with operands of mixed types, C++ *promotes* the types of values to do the calculations

- *promotes* means converts to a more precise type

For example, for expression `x/y`

- If `x` is of type `int` and `y` is of type `float`
  - x's value is read, converted to a float and then used in division. The result is float.
- If x is of type `float` and y is of type `double`
  - x's value is read, converted to a double and then used in division. The result is double.

36

Exercise: Expression Types

```
int x = 5, y = 2;
float f = 2.0;
double d = 5;
```

What is the result of "**x/y**"?

- What is the type of "**x/y**"?

What is the result of "**x/f**"?

- What is the type of "**x/f**"?

What is the result of "**d/f**"?

- What is the type of "**d/f**"?

37

Explicit Conversions (Type Casting)

We can also explicitly change type

Type cast operator: (*type-name*) *operand*

```
int A = 9, B = 2;
float  C;

C = A / B;          /* C is 4.0 */

C = A /(float)B;   /* C is 4.5*/
```

Doesn't change the value of B, just changes the type to float

38

Outline

Variables

Data types

Constants

Expressions

Assignment statements

I/O statements

Lab Exercises

39

Assignment Statements

An assignment statement changes the value of a variable

Format:

```
variable = expression;
```

- Assignment statements end with a semi-colon
- The single variable to be changed is always on the left of the assignment operator '**=**'
- On the right of the assignment operator can be a
  - Constant
    ```
    age = 21;
    ```
  - Variable
    ```
    my_cost = your_cost;
    ```
  - Expression
    ```
    circumference = diameter * 3.14159;
    ```

40

Assignment Statements and Algebra

The '=' operator in C++ is not an equal sign

- The following statement cannot be true in algebra

```
number_of_bars = number_of_bars  +  3;
```

- In C++, it means the new value of **number_of_bars** is the previous value of **number_of_bars** plus 3

41

Type Conversions across Assignments

It is better that

- the type of the variable on the left of assignment operator '=' is the same as the type of the expression on the right.

But C++ allows the types to be different

The value of the right side is converted to the type of the left, and then assigned to the variable on the left

```
int i = 512;
double x;
x = i;  /*value of i is converted to double*/
```

42

Type Conversions across Assignments

If the left side is of smaller range or precision, information may be lost (**should avoid**)

- **`float`** to **`int`** truncates any fractional part.

```
float f = 123.6;
int i;
i = f;
```
(*what is the value of* **`i`**?)

- If the value of float is out of the **`int`** range, **`float`** to **`int`** results in strange value.

```
float f = 2e34;
int i;
i = f ;
```
(*what is value of* **`i`**?)

43

Type Conversion - Examples

```
int x=5, y=2, w;
double z, q = 2;

w = x/y;
      // w = 2
z = x/y;
      // z = 2.0
z = x/q;
      // z = 2.5
w = x/q;
      // w = 2
```

44

char ← → int

The following actions are possible but generally not recommended!

It is possible to store char values in integer variables

```
int v = 'A';
```

variable **`v`** will contain an integer (65) representing 'A'

It is possible to store int values in char variables

```
char letter = 65;
cout << letter;
```

This will print **`A`** on the screen.

45

bool ← → int

The following actions are possible but generally not recommended!

Values of type **`bool`** can be assigned to **`int`** variables

- True is stored as 1
- False is stored as 0

```
bool b=true;
int i;
i = b;
cout << b;   // This will output 1
```

Values of type int can be assigned to bool variables

- Any non-zero integer is stored as true
- Zero is stored as false

46

Shorthand Assignment Operators

Some expressions occur so often that C++ contains to ***shorthand operators*** for them

All arithmetic operators can be used this way

- **`+=`**
  `count += 2;` means **`count = count + 2;`**
- **`*=`**
  `bonus *= 2;` means **`bonus = bonus * 2;`**
- **`/=`**
  `x /= y;` means **`x = x / y;`**
- **`%=`**
  `rem %= (cnt1 + cnt2);` means
  `rem = rem % (cnt1+ cnt2);`

47

Outline

Variables
Data types
Constants
Expressions
Assignment statements
I/O statements
Lab exercises

48

Input and Output

A data stream is a sequence of data
- Typically in the form of characters or numbers

An input stream is data for the program to use
- Typically originates
  - at the keyboard (standard input)
  - at a file

An output stream is the program's output
- Destination is typically
  - the monitor (standard output)
  - a file

Output using cout

cout is an output stream sending data to the monitor

The insertion operator "**<<**" inserts data into **cout**

Example:

```
cout << number_of_bars << " candy bars\n";
```

- This line sends two items to the monitor
  - The value of **number_of_bars**
  - The quoted string of characters **" candy bars\n"**
    - Notice the *space* before the '**c**' in **candy**
    - The '**\n**' causes a new line to be started following the '**s**' in **bars**
  - A new insertion operator is used for each item of output

Examples Using cout

This produces the same result as the previous sample

```
cout << number_of_bars ;
cout << " candy bars\n";
```

Below arithmetic is performed in the **cout** statement:

```
cout << "Total cost is $" << (price + tax);
```

Quoted strings are enclosed in double quotes ("Walter")
- Don't use two single quotes (')

A blank space can also be inserted with

```
cout << " " ;
```

if you do not put a string in the front of **"candy bars\n"**

Escape Sequences

Escape sequences tell the compiler to treat characters in a special way

'\' is the escape character
- To create a newline in output use **\n** like in:

```
cout << "\n";
```

or the newer alternative:

```
cout << endl;
```

Example:
```
cout << sum << "\n";
```
is the same as:
```
cout << sum << endl;
```

- Other escape sequences:

\t -- a tab
**** -- a backslash character
\" -- a quote character

Include Directives

The **include** directives add library files to our programs
- To make the definitions of the **cin** and **cout** available to the program:

```
#include <iostream>
```

The **using** directives include a collection of defined names
- To make the names **cin** and **cout** available to the program:

```
using namespace std;
```


Example 1

Write a C++ program that outputs:

```
  XXX
 X   X
X
X
X
X
 X   X
  XXX
```

This 4 lines can be replaced with
```
cout << "X\nX\nX\nX\nX\n";
```

Program:

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

int main()
{
  cout << "  XXX\n";
  cout << " X   X\n";
  cout << "X\n";
  cout << "X\n";
  cout << "X\n";
  cout << "X\n";
  cout << " X   X\n";
  cout << "  XXX\n";

  return 0;
}
```

Example 2 of Escape Sequences

Writing a C++ program which outputs the following line to the screen:

```
Use \n to insert a newline in cout!
```

Program:

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

int main()
{
  cout << "Use \\n to insert a newline in cout!\n";

  return 0;
}
```

55

Example 3 of Escape Sequences

Writing a C++ program which outputs the following line to the screen:

```
Say "Hello" to the nice people!
```

Program:

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

int main()
{
  cout << "Say \"Hello\" to the nice people!\n";

  return 0;
}
```

56

Formatting the Output of Real Numbers

Real numbers (such as type double) can be displayed in a variety of formats

- If the absolute value of the number is not too small or too big, it is displayed in the fixed-point format (ordinary notation)

```
double price = 78.5;
cout << "The price is $" << price << endl;
```

The output is:

```
The price is $78.5
```

- If it is too big or small, it is displayed in the e-notation:

```
double price = 1234567.12;
cout << "The price is $" << price << endl;
```

The output is:

```
The price is $1.23457e+06
```

57

Formatting the Output of Real Numbers

Sometimes, we would like to output the real numbers with certain format and precision

- E.g., we would like to control the number of decimal places displayed.

For example,

```
double price = 78.5;
cout << "The price is $" << price << endl;
```

- The default output of the above is:

```
The price is $78.5
```

- But preferred format of output for money amount is :

```
The price is $78.50
```

58

Showing Decimal Places

`cout` includes tools to specify the output of type double

To specify fixed point notation

- **`setf(ios::fixed)`**

To specify that the decimal point will always be shown

- **`setf(ios::showpoint)`**

To specify that two decimal places will always be shown

- **`precision(2)`**

Example:

```
cout.setf(ios::fixed);
cout.setf(ios::showpoint);
cout.precision(2);
cout << "The price is $" << price << endl;
```

59

Input Using cin

`cin` is an input stream bringing data from the keyboard

The extraction operator (**`>>`**) removes data to be used

Example:

```
cout << "Enter the number of bars in a package\n";
cout << " and the weight in ounces of one bar:\n";
cin >> number_of_bars;
cin >> one_weight;
```

This code prompts the user to enter data then reads two data items from **`cin`**

- The first value read is stored in **`number_of_bars`**
- The second value read is stored in **`one_weight`**
- Data is separated by white spaces (space, newline or tab) when entered

60

Reading Data From cin

Multiple data items are separated by white spaces

Data is not read until the enter key is pressed

- Allows user to make corrections

Example:

```
cin >> v1 >> v2 >> v3;
```

- Requires three white space separated values
- User might type:

```
34  45  12   <enter key>
```

or

```
34
45          12   <enter key>
```

61

Outline

Variables

Data types

Constants

Expressions

Assignment statements

I/O statements

Lab exercises

62

Exercise 1

Write a C++ program named **`lab2ex1.cpp`** that

- prompts the user to input a value for real-valued variable *x*. The prompt message is "**`Please enter the value for x:`**"
- computes the value for *y* given the inputted value for *x*, where $y=2x^2+1$
- outputs to the screen "**`The y value is`**", followed by the calculated y value.

63

Exercise 2

Write a C++ program named **`lab2ex2.cpp`** that

- prompts the user to enter the scores for the first, second and third games. The scores are whole numbers and thus should be stored in integer variables.
- calculates the average score
- outputs "**`The average score is:`**", followed by the calculated average score, shown with two decimal digits after the decimal points.

64

Exercise 3

Write a C++ program named **`lab2ex3.cpp`** that

- prompts the user to enter a distance in miles
- converts the distance value into kilometers. Note that 1 mile equals to 1.61 kilometers.
- outputs the following to the screen:

```
m miles is k kilometers
```

where **`m`** is the miles value the user inputted and **`k`** is the converted kilometer value. Both values should have one digit after the decimal point**.** For example, if the user inputs 8, your program should display the following to the screen:

```
8.0 miles is 12.9 kilometers
```

65

Exercise 4

Write a C++ program named **`lab2ex4.cpp`** that

- prompts the user to enter a number of quarters, dimes, and nickels
- outputs the monetary value of the coins in dollars.
- For example, if the user enters 2 for the number of quarters, 3 for the number of dimes, and 1 for the number of nickels, then the program should output:

```
The coins are worth $0.85.
```

Note that the value should be shown using ordinary notation and with two decimal places.

66

Exercise 5

Write a C++ program named **`lab2ex5.cpp`** that

- prompts the user to enter a time in seconds
- outputs how far an object would drop if it is in free fall for that length of time
- Assume that the object starts at rest, there is no friction or resistance from air, and there is a *constant acceleration of 32 feet per second* due to gravity. Use the equation:

$$distance = \frac{acceleration \times time^2}{2}$$

Please use a declared constant for *acceleration*.

- For example, if the user enters 9 for the length of time, your program should output:

```
The object would drop 1296 feet during 9 seconds.
```

67

Exercise 6

Write a C++ program named **`lab2ex6.cpp`** that

- See Question 14 on Page 59 in the textbook

68

How to submit your programs

Two ways:

- Use the submit command on Linux command line

```
submit 5910 lab2 lab2ex3.cpp
```

This will submit your file **`lab2ex3.cpp`** to the lab2 directory under

`/cse/dept/course/2012-13/F/5910/submit/`

- Submit through the web site:

https://webapp.cse.yorku.ca/submit/

Please choose the correct course and lab ids to submit your files.

69