CEE432/CEE532/MAE541
Developing Software for Engineering Applications

Lecture 2: Components of a Simple C++ Program

Common Components

- **Preprocessing directives**: Tell the compiler what needs to be done.
- **Variables**: Store “values”.
- **Expressions**: Are made up of variables, constants, operators etc.
- **Statements**: A statement is made up of one or more of the following - variables, expressions, C++ keywords and tokens.
- **Input and Output**: Obtain “values” from keyboard, file etc. and display “values” to the screen, file etc.
- **Functions**: Program component that can be called from other parts of the program.

C++ Rules

- All programs must have a **main** function
- This function must return an integer value
- The statements in the main program are contained within the two braces

```
{
…. .
}
```

- Statements are normally executed sequentially

The Simplest C++ Program

```
#include <iostream>

int main ()
{
  std::cout << “Hello World”;
  return (0);
}
```

Reserved C++ Keywords

```
auto break case char const continue default
  do double else enum extern float for goto
  if int long register return short signed
  sizeof static struct switch typedef union
  unsigned void volatile while

asm bool catch class const_cast delete
  dynamic_cast explicit export false friend
  inline mutable namespace new operator
  private protected public reintrepret_cast
  static_cast template this throw true try
  typeid typename using virtual wchar_t
```

C++ Tokens

```
# ;  { } “ ” \ / * : <<
? () >> ! >= <= []
~ :: -> ++ -- & && ||
```

Variables and Data Types

- Built-in data types
  - integers
  - real
  - boolean
  - characters
- Scalar versus array data types
- User-defined data types

Bits, Bytes and Words

- A **bit** is the smallest data storage element (it can store either a 0 or a 1 value)
- A **byte** is a collection of 8 bits
- A **word** depends on the hardware. Some machines have 32-bit word length. Increasingly 64-bit word machines are becoming commonplace.

Constants

- Integer (**`-34, 904, 5609L`**)
- Single precision (**`-1.03f, 0.006f`**)
- Double precision (**`-1.03, 0.001e-5`**)
- Boolean (**`true`** or **`false`**)
- Character (**`‘X’, ‘v’`**)

Data Types

Use **`#include <limits>`**

Data Type	Values (System dependent)	C++ Example
`short`	16 bits long. Integer value from -32767 to 32767	`-145`
`int`	32 bits long. Integer value from -2 147 483 648 to 2 147 483 647	`1034`
`long`	64 bits long. Integer value from –9 223 372 036 854 775 808 to 9 223 372 036 854 775 807	`80967845L`
`float`	32 bits long. Floating point value in the range $\pm 3.4(10^{\pm 38})$ (7 digits precision)	`-0.0035f`
`double`	64 bits long. Floating point value in the range $\pm 1.7(10^{\pm 308})$ (15 digits precision)	`1.45`
`bool`	false or true	`true`
`char`	1 character	`g`

Variables

- Naming variables
  - A variable name can have many characters, starts with an alphabet, and typically is made up of the following characters – **`a`** through **`z`**, **`A`** through **`Z`**, **`0`** through **`9`**, and _. No blank spaces are allowed.
- Variable names are case sensitive

Declaring Variables

```
int nHours;
int nHours, nMinutes, nSeconds;
int nHours; int nMinutes;
float fX, fY;
float fCoordinates[2];
char cInput;
char szName[11];
bool bHappy;
```

Defining Variables

```
int nHours=10, nMinutes = 20;
float fWidth = 1.23f;
double dError = 1.0e-4;
bool bHappy = true;
char cID = 'X';
int nVX[3] = {1, -5, 6};
```

Using Comments

- Single line comment

```
int nCount = 10; // number of people
```

- Block comment

```
/* Calculates the area of a circle.
Input to this function is the radius.
Output from this function is the area. */
```

Mathematical Operators

OPERATOR	MEANING
+, -	Unary positive, negative
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (or remainder)

Operator Precedence

- **Ascending order**

```
0. =        Assignment
1. +        Addition
1. -        Subtraction
2. *        Multiplication
2. /        Division
2. %        Modulus
3. ()       Parenthesis
4. +, -     Unary positive, negative
```

Mathematical Functions

Accessed via

```
#include <cmath>
```

```
sin(x)      sqrt(x)
cos(x)      pow(x,y)
tan(x)      log(x)
asin(x)     log10(x)
acos(x)     abs(x)
atan(x)     fabs(x)
```

Angles are in radians

Expression Evaluation

- Use operator precedence
- Evaluate left to right
- Use parenthesis to make the operation clear

Example Expressions

Expression	Order of evaluation	Evaluates to
5+3-2	5+3=8-2=6	6
5+3*2-1	3*2=6;5+6=11-1=10	10
(5+3)*2-1	5+3=8;8*2=16-1=15	15
5*6/3	5*6=30/3=10	10
5*(6/4)	6/4=1;5*1=5	5
5*6/4	5*6=30/4=7	7
5%2*3	5%2=1*3=3	3

Assignment Statement

```
variable = expression
variable = variable operator expression
```

Examples

```
nHits = 2*nSaves - 10;
```

```
dArea = 3.1415926*pow(dRadius,2.0);
```

- Be careful with mixed data type arithmetic

Assignment Statement

Assignment Operator	Sample C++ statement	Equivalent C++ statement
+=	c += 9;	c = c + 9;
-=	d -= 5;	d = d – 5;
*=	e *= 2;	e = e * 2;
/=	f /= 3;	f = f / 3;
%=	g %= 7;	g = g % 7;

Mathematical Expression

Mathematical expression	C++ expression
$\frac{bh^3}{12}$	`fB*pow(fH, 3.0)/12.0`
$\frac{ML^2}{2EI}$	`(fM*fL*fL)/(2.0*fE*fI)`
$\frac{\sin(a)\cos(b)}{\sqrt{1+c^2}} - 5.5$	`(sin(fA)*cos(fB))/sqrt(1.0+fC*fC)-5.5`
$y\lvert x - \log(a)\rvert$	`fY*fabs(fX-log(fA))`
$10t\exp(-t)$	`10.0*fT*exp(-fT)`
$\frac{1}{2} + \sin^{-1}\left(\frac{a}{\pi}\right)$	`0.5+asin(fA/3.1415926)`

Input and Output

- Stream: A sequence of bytes
- Input: Bytes flow from an input device (e.g. keyboard) to main memory. **`istream class`**
- Output: Bytes flow from main memory to an output device (e.g. screen). **`Ostream class`**
- Need to use the **`iostream`** header file. This supports both input and output.

```
#include <iostream>
```

Simple Input

- Need to use stream-extraction operator **>>**
- Input by default is from the standard input device
- To read an integer

```
int nScore;

cin >> nScore;
```

- Blank spaces, tabs, newline characters are skipped.

Simple Output

- Need to use stream-insertion operator **<<**
- Output by default is to the standard output device
- To display an integer

```
int nScore;

cout << nScore;
```

- To display a character string

```
cout << “Your score is “ << nScore <<
  ‘\n’;
```

Formatting Output

- Floating-Point Precision
  - setprecision() controls the number of significant digits or significant decimal digits with one integer argument
  - precision() same as above and returns the current precision setting with no argument (precision 0 restores the default precision value 6 for both)
- Field Width
  - width() sets the field width and returns the previous width with one integer argument, and with no argument returns the current setting
  - setw() sets field width (a value wider than the field width will not be truncated and width setting applies only for the next insertion or extraction)
- Requires **`#include <iomanip>`**

Using Vector Variables

- Needed if more than one value needs to be stored (next to each other)
- Defined as follows

```
datatype variablename[integer constant
  that is the size];
```

- Here is an example

```
double dVX[3];
dVX[0]=1.1; dVX[1]=1.2; dVX[2]=1.3;
```

Character String

```
char szName[5]="Alan"; // defined and initialized
szName="Alan";         // invalid
szName[0] ='A';
szName[1] ='l';
szName[2] ='a';

szName[3] ='n';
```

Formatted Output

- Look at all examples in Chapter 2
- Write a program for Problem 2.9
- Learn to use the debugger

Summary

- Definitions and declarations must precede usage. Look at C++ compiler having access to a program dictionary that has three components – (a) C++ keywords and tokens, (b) functions whose prototypes are available in header files, and (c) user-defined variables and functions. Imagine that this is a dynamic dictionary with respect to (c). In other words, items may be added to the dictionary while the compiler is interpreting the program. If something is used in the program that cannot be found in the dictionary, then the compiler issues an error message.

Summary

- Every program has one and only one main function. This is the location from where the execution of the program starts.
- Comments in programs start with the pair /* and end with the pair */. Comments on a single line occur to the right of the pair //.
- A block of statements occur within the braces { and }. As an example, the statements in the main function can be found within the braces. A semicolon ; is used to terminate most commonly used C++ statements.

Summary

- Usually there are two types of files found in C++ programs – .cpp and .h files. Usually the .cpp files contain the statements that are executed, and the .h files contain the definitions and declarations. Note that the file extensions may be different with different compilers and IDEs.
- The more commonly used standard data types are short, int, long, float, double, bool and char

Summary

- Variables are used to store values of the standard data types. Variables are identified by variable names.
- Scalar variables store one value. Vector variables store several values.
- Functions are independent program segments that can be called from different locations in a program.

Summary

- C++ provides several functions including mathematical that the programmer can use in his or her program.
- Expressions can be created using constants, variables, functions and operators. Expressions with mathematical operators follow certain evaluation rules.

Summary

- There are several types of C++ statements such as assignment, input/output, etc.
- C++ provides streams (sequence of bytes) for obtaining input from devices such as a keyboard or disk, and for outputting data to a display device, printer, or disk.