Getting Ready for Canadian Computing Competition

Class 1
November 30, 2019

Easy Path背景介绍

给大家简单地介绍一下我们易路公司Easy Path的背景。我们公司由加拿大易维教育 集团(Easy Group)和美国 PanoPath过来人共同成立。得益于两家公司在北美留学市场的多年积累，我们现在拥有超过1000名北美本地导师。因为了解留学生活的彷徨、艰辛和迷茫，所以我们秉承着“从留学生中来，到留学生中去”的理念。凭借着强大的导师资源和百分之百的责任心，我们相信可以帮助和你一样的留学生在求学路上轻松前行，把留学路变成真正的易路。

答题器：

请选择正确描写自己C++基础的选项

A. 无，但是上过 AP CS
B. 无，但是有其它编程语言基础
C. 无，也没有其他编程语言基础
D. 有基础，写过200行及以上的program

小黑板

你现在在哪里上课？（比如哪个国家，哪个省）

什么是CCC

- 分为Junior and Senior两个级别
- 考试日期Feb. 12, 2020
- 一共有三小时，回答五个问题
- 满分75分，每道题15分
- 每个问题最多能回答50次，后台会取分数最高的那一次
- 竞赛中可以使用reference materials

CCC 题目难度分布

Questions 1 and 2	**Basic algorithms (e.g., sorting, searching)**
Questions 3 and 4	More advanced algorithms (e.g., careful counting, some mathematical reasoning)
Question 5	IOI level question

CCC 课程安排 – 编程课 (1.5 x 8 小时)

Class	Date	Content
		Variables, types, and constants
		Operators
		Control Flow Statements (if, continue, break)
1	Nov. 30, 2019	standard I/O and cerr
		Functions, function overloading, scopes
		Loops
2	Dec. 8, 2019	Arrays
		Structures
3	Dec. 14, 2019	Sorting algorithms (insertion sort, bubble sort, merge sort, etc.)
		Pointers
4	Dec. 21, 2019	Recursion
		Pointers, lvalue referencing
5	Dec. 28, 2019	Passing by value vs passing by reference
6	Jan. 4, 2019	Trees
7	Jan. 11, 2019	STL: Stack, queue, vector, 2D vectors
		call stack
8	Jan. 18, 2019	debugging, how to use the debugger, sample CCC problem

Outline

- Code::blocks IDE and C++14 Compiler
- Variables
  - Types
  - Declaration
  - Assignment operator
  - Arithmetic operations
- Control flow statements
  - If … else
- I/O

Code::blocks IDE

- IDE – integrated development environment
- Compiler – GNU G++ / GCC
  - A compiler translates your code to machine language
  - The translation begins from top to bottom
- Errors
  - Syntax – “grammar” of the programming language
  - If you cannot compile your code, you have a **compilation error**, or a **syntax error**
  - Once your code is compiled, it will start to execute/run. If an error occurs, we call it a **runtime error** or an **execution error**

“Hello world” Program

“Hello world” Program

```
#include <iostream> // for I/O
using namespace std; // always include

int main() { // main function declaration
    cout << "Hello world!" << endl; // output line
    return 0; // return value, matches the function declaration
}
```

“Hello world” Program

```
#include <iostream> // for I/O
using namespace std; // always include

int main() { // main function declaration
    cout << "Hello world!" << endl; // output line
    return 0; // return value, matches the function declaration
}
```

When we click “build and run” . . .

Variables

- Values (1, 2, ‘A’, “hello world”)
- Variables store any type of values
- Values can be changed during execution
- **Variables** are the name of memory locations allocated by compilers
  - Compiler must know how much space to allocate because this size will not change
  - We tell compilers this information by declaring variables
  - A declaration includes:

```
datatype variable_name assignment_operator= value;
```

Example

```
datatype variable_name assignment_operator= value;
```

```
int a = 20;
```

Variable_name -> A

Value -> B

```
Datatype -> C
```

```
Assignment operator -> D
```

Example

```
datatype variable_name assignment_operator= value;
```

```
int a = 20;
```

Variable_name -> A

Value -> B

Datatype -> C

Assignment operator -> D

What is A, B, C, D:

A. 20, a, int, =

B. int, 20, a, =

C. =, a, 20, int

D. a, 20, int, =

Example Answer

```
datatype variable_name assignment_operator= value;
```

```
int a = 20;
```

Variable_name -> a

Value -> 20

```
Datatype -> int
```

```
Assignment operator -> =
```

Assignment Operator

- LHS = RHS
- It assigns the value of the RHS to the variable on the LHS
- The LHS is also known as lvalue, the RHS is the rvalue
  - Ex). If we had a variable x that is an integer, we can increment it by 1:

    x = x + 1;

    Or

    x += 1

    Or

    x++; // this is where the name c++ came from!
  - Of course there are -=, *=, /=, x--, --x, ++x
  - Note the difference between ++x and x++
- *Note: If you want to compare two strings, use ==
  - Even though 3 = 3, it is as wrong as 3 = 4!

Example

- What does the following print?

```
int k = 10, l = 10, m = 10, n = 10;
c = k++;
cout << c << " " << k << endl;
c = l--;
cout << c << " " << l << endl;
c = ++m;
cout << c << " " << m << endl;
c = --n;
cout << c << " " << n << endl;
```

Example Solution

- What does the following print?

```
int k = 10, l = 10, m = 10, n = 10;
c = k++;
cout << c << " " << k << endl;
c = l--;
cout << c << " " << l << endl;
c = ++m;
cout << c << " " << m << endl;
c = --n;
cout << c << " " << n << endl;
```

```
10 11
10 9
11 11
9 9
```

Variable Name

- All variable names must:
  - Begin with a letter of the alphabet or an underscore(_)
  - After the initial letter, it can contain letters and numbers.
  - No spaces or special characters
- You cannot use a C++ keyword as a variable name, like cout, endl
- It is case sensitive, so name is not the same as Name
- Since there are no spaces, you can use either camelcase or underscore:
  - Ex). myVariable -> camelCase
    my_variable -> underscore

Exercise

Which of the following are valid variable names?

A. 1Name

B. Variable_Name

C. var name

D. $varname

Datatypes

String

- A string of characters of arbitrary length
- must use double quotes

Ex). “Hello world!”

Declared as: string x;

Declaration with initialization: string x = “”; // initialized to an empty string

Datatypes

String

- A string of characters of arbitrary length
- must use double quotes

Ex). “Hello world!”

Declared as: string x;

Declaration with initialization: string x = “”; // initialized to an empty string

Which one of the following is a valid string declaration?

A. string name = ‘Jasmine’;

B. string number = “14”;

C. string name = Jasmine

D. String name = “”;

Datatypes

Integers

- An integer with no decimal points

Ex). int x = 10;

Double vs Float – storing numbers with decimal point

- Float - 32 bit
- float x = 1.2;
- Double - 64 bit
- double x = 1.2;

Datatypes

Characters

- One single quote
- Ex). char x = ‘A’;

  ‘\n’ – newline

  ‘\t’ – tab

Datatypes

Booleans

- True(1, not zero) or false(0)
- Ex). There are all valid declaration and initializations:

```
bool  n = 0;
bool  n = false;
bool  n = 1;
bool  n = true;
bool n = 20; // true, not zero
```

Exercise

- Which one of the following would not compile?

A. int x = 1000000;

B. integer y = 3;

C. string = “abc”;

D. int z = “abc”;

Exercise

- Which one of the following would not compile?

A. bool switch = True;

B. int x = 1

C. char a = ‘AB’;

D. string my_age = “one hundred”;

Exercise

- Inside the main function, declare a string and an int, then use cout to print information about you. Like “I ate 10 apples yesterday”, or “I am in grade 12!”

```
#include <iostream>

using namespace std;

int main() {

    // your code goes here

};
```

Arithmetic Operations

Operator	Description	Example
+	Adds two operands.	A + B = 30
-	Subtracts second operand from the first.	A - B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0

/ operator
always performs integer division (floor)
Ex).
3/2 gives 1
7/4 gives 1

* 7/(double)4 this gives you 1.75

% operator
Useful for finding divisors.

Exercise

- What is the output of this code?

```
int a = 21;
int b = 10;
int c ;
c = a + b;
cout << c << endl;
c = a - b;
cout << c << endl;
c = a * b;
cout << c << endl;
c = a / b;
cout << c << endl;
c = a % b;
cout << c << endl;
```

Exercise

- What is the output of this code?

```
int a = 21;
int b = 10;
int c ;
c = a + b;
cout << c << endl;
c = a - b;
cout << c << endl;
c = a * b;
cout << c << endl;
c = a / b;
cout << c << endl;
c = a % b;
cout << c << endl;
```

```
31

11

210

2

1
```

Exercise

Decompose a three digit number, num.

Sample input:

123

Sample Output:

1 * 100 + 2 * 10 + 3

I/O – cin

We import the library

```
#include <iostream>
int main () {
        int a; // declare a variable for cin to read the data
        cin >> a; // will read the longest integer possible
        cout << a; // print out what we have just read
}
```

If we input a

If we input 12a

I/O – cin

We import the library

```
#include <iostream>
int main () {
        int a; // declare a variable for cin to read the data
        cin >> a; // will read the longest integer possible
        cout << a; // print out what we have just read
}
```

If we input a -> nothing is read in

If we input 12a -> assign a to 12

I/O - cin

- cin will read the longest chain of characters before making the datatype invalid
- This will ensure that the input of 123 as an int will read entirely

- Ex). Nothing makes a string invalid, it will read everything until next line or EOF

  char type: it will only read in one char, but nothing makes it fail

  double/float: 6 digits, trim the rest

I/O – cin

- cin is also a function that returns a value
  - It returns a bool (true/false)
  - It returns true if the read was successful, false otherwise
- Ex). We can combine this with an if … else statement later

I/O – cin

- EOF – end of file
- Ctrl + Z -> windows
- Ctrl + D -> Linux
- Shows that you have finished input
- When you change your input to an input file, EOF is automatically added at the end

Ex). Now for the code above, we can enter

123^Z to read the line and terminate the program

This will be useful later when we try to use loops to read input

Cerr

- For debugging purposes
- It does not print to cout, but you can see it with all other cout results
- The server will only check your output stream when checking the solution

cout

- Handles string, int, double, float, bool, char
- You can print multiple types in one line
- You can print the values themselves or variables of the values
- endl starts a new line
  - Ex). cout << endl; // prints a blank line

If ... Else

- Syntax:

```
if (Boolean_Expression_1) {
  Action1();
} else if (Boolean_Expression_2) {
  Action2();
} else {
  Action3();
}
```

- Braces can be omitted if there is only one statement (not recommended)
- “else” is optional
- Statements can be nested

More Operators

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.

Operator	Description	Example
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

! not
&& and
|| or

Example

For the cin problem, we want to read in an integer. If it fails, print “wrong input!”. Otherwise, print the integer read int.

Example

For the cin problem, we want to read in an integer. If it fails, print “wrong input!”. Otherwise, print the integer read int.

```
int main (){
    int num;
    if (!(cin >> num)){
        cout << "wrong input!" << endl;
    }
    else {
        cout << num;
    }
}
```

Exercise

Read in two integers a, b, separated by a space.

Print the larger integer using <, >, and if… else. If they are equivalent, print either.

Sample input:

1 10

Samle output:

10

Understanding Nested if and the Flow

Given the pseudocode below, write it in one if… else statement

```
if (I am in class){
      if (I fell asleep) {
         learned nothing
      }
      else {
         learned everything
      }
   }
   else if (I am a genious){
      learned everything
   }
   else {
      learned nothing
   }
```