WEEK 1: INTRO TO C++

SEPTEMBER 17, 2016

COMPUTER SCIENCE ENRICHMENT CLUB

WHY C++?

- Performance – fast language with high abstraction
- Macros make code smaller
- Rich library

IDE – CODE::BLOCKS

- Create Project
  - File -> New -> Project->Console Application
- Create file
  - File -> New -> Empty File

- Download from - http://www.codeblocks.org/downloads
- WARNING: Code::Blocks might not be available in ICPC system

TAKE AWAY

- Logical thinking
- Write code and Syntax
- Tackling basic problems
- Core concepts of C++ & most programming languages

LIBRARIES

- <iostream> - print to console
- <string> - string data type
- <vector> - dynamic array
- <queue>, <list>, <stack>, <set> - data structure(s)
- <algorithm> - sort
- <math> - math functions (sin, cos) and values like MATH.PI

OUTPUT

- First of all, code is read by the computer top down

- We output to the console using the command –

```
std::cout << " " ;
```

- Notice the arrow signs it is pointing towards the command. This will be important later

“HELLO WORLD”

```
1  #include<bits/stdc++.h>
2
3  int main(){
4      std::cout << "Hello World\n";
5
6      return 0;
7  }
```

FASTER “HELLO WORLD”

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  int main(){
6      cout << "Hello World\n";
7
8      return 0;
9  }
```

COMMENTING

```
int main() {

    /* This is a multi line comment
    what I write here does not compile
    */
    // this is a singlie line comment

    return 0;
}
```

VARIABLES

- Variable is an item that can take one or more value

- **Boolean** – True/False
- **Integer** – number without decimal
- **Double** – decimal number
- **Char** – holds letter

DATA TYPES

```
int main(){
    // primitive data types c++
    bool status = false;

    int int_32 = 2+5;          // 32 bit
    long long int_64 = 21;     // 64 bit

    char character = 'a';

    float decimal_32;           // 32 bit
    decimal_32 = 3.2;          // to show var does not
                               //have to be initialized at declaration
    double decimal_64 = 5.5;     // 64 bit

    return 0;
}
```

STRINGS

- String is a data type/ variable that holds sequence of characters.
- Characters range everything from numbers to symbols
- Ex – “H311o WoRID!!!”

STRINGS

```
int main(){
    int length;

    string word = "Hello World";
    length = word.length();

    cout << word << "\n";
    cout << "The length of the word is: " << length << "\n";

    return 0;

    /*Output
    Hello World
    11
    */
}
```

INPUT

- Sometimes we like to give the user the option to interact more with the program
- The user can enter a string, int, double data types
- The command to do this is: cin >> [var];
  - Notice how the arrows go towards the variable instead of the command unlike cout

USER INPUT

```
int main(){
    string first_name, last_name;
    int age;

    cout << "Enter your first name: ";
    cin >> first_name;
    cout << "Enter your last name and age: ";
    cin >> last_name >> age;

    cout << "Hello " << first_name << " " << last_name << "\n";
    cout << "You are " << age << " years old\n";

    return 0;

}
```

```
/*
Input:
Bill
Nye 60
Output:
Hello Bill Nye
You are 60 years old
*/
```

BREAK

CHALLENGE

Take in two numbers. Return their product.

HINTS:

```
int x = 0;              // DECLARING VARIABLES
cin >> x;               // GETTING INPUT
cout << x;              // PRINTING OUTPUT
```

BREAK – SOLUTION

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


int main(){
	int x, y;
	int product;
	cin >> x >> y;
	product = x * y;
	cout << product << “\n”;
}
```

FUNCTIONS

- Functions are a group of commands which work together to do a specific task
- Makes code more readable
  - Gets rid of repeated code
- Depending on the design of the function, it may or may not return a value back

FUNCTIONS

```
/*
Input:
Bill
Alan
60
Output:
Bill
*/
```

```
int get_age(){
    int age;
    cout << "Enter your age: ";
    cin >> age;
    return age;
}

string get_name(){
    string name;
    cout << "Enter your name: ";
    cin >> name;
    return name;
}

void print(string input){
    cout << input << "\n";
}

int main (){
    string name1, name2;
    int age;

    name1 = get_name(); name2 = get_name();age = get_age();
    print(name1);

    return 0;
}
```

FUNCTIONS OPTIONAL PARAMETERS

```
int add(int x, int y = 0){
    return x + y;
}

int main(){
    int number1 = add(2);
    int number2 = add(2,2);

    cout << number1 << "\n";
    cout << number2 << "\n";

    return 0;
}
```

```
/*
Output:
2
4
*/
```

FORWARD DECLARATIONS FUNCTIONS

- Code is read top down
- Forward declarations tells the compiler at runtime that a function with that name and parameters exists
  - This way a function can be called even if it much lower in the code than the point it was declared at

FORWARD DECLARATIONS FUNCTIONS

```
void print(string);

int main(){
    string name = "Charles Babbage";

    print(name);

    return 0;
}

void print(string name){
    cout << name << "\n";
}
```

```
/*
Output:
Charles Babbage
*/
```

LOCAL VS GLOBAL

- Global variables are available to the entire file
- Local variables are available only to the function it is declared in

LOCAL VS GLOBAL

```
// these are global declarations
using namespace std;
int counter = 0;

void increase_counter(){
    // name var not available here unless sent as parameter
    // counter var available here
    counter = counter +1;
}

int main(){
    // this is a local declaration
    string name = "Steve Wozniak";
    // counter var available here

    cout << "My name is " << name << "\n";

    increase_counter();increase_counter();increase_counter();
    cout << counter << "\n";

    return 0;
}
```

```
/*
Output:
My name is Steve Wozniak
3
*/
```

MATH

- Add: +
- Subtract: -
- Multiply: *
- Divide: /
- Mod: %

MATH

```cpp
int main(){
    int add, subtract;
    float divide, multiply, mod;

    add = 2+5;
    add += 3;
    cout << "Add: " << add << "\n";

    subtract = 2+5;
    subtract -= 3;
    cout << "Subtract: " << subtract << "\n";

    divide = 5/3;
    divide /= 2;
    cout << "Divide: " << divide << "\n";

    multiply = 2*5;
    multiply *= 3;
    cout << "Multiply: " << multiply << "\n";

    mod = 5%3;
    cout << "Mod: " << mod << "\n";

    return 0;
}
```

```
/*
Output:
Add: 10
Subtract: 4
Divide: 0.5
Multiply: 30
Mod: 2
*/
```

INCREMENT

```
int main(){
    int x = 0;

    cout << x << "\n";        // print 0

    cout << x++ << "\n";      // print 0

    cout << x << "\n";        // print 1

    cout << ++x << "\n";      // print 2

    cout << x-- << "\n";      // print 2

    cout << x << "\n";        // print 1

    return 0;
}
```

LOGIC

>	Greater than
<	Less than
>=	Greater than or equals to
<=	Less than or equals to
==	equals to. Don't confuse with =
!=	not equals to

LOGIC

```
int main(){
    // 0 == False
    // 1 == True

    cout << (2<3) << "\n";   // print 1

    cout << (2>3) << "\n";   // print 0

    cout << (3==2) << "\n"; // print 0

    cout << (3!=2) << "\n"; // print 1

    return 0;
}
```

TYPE CONVERSION

```
int main() {
    // only works between primitive data types
    char letter = 'a';

    //converts to ascii
    int num = int(letter);

    cout << num << "\n";

    double decimal = double(num);

    // to show conversion happened
    decimal += 5.5;

    cout << decimal << "\n";

    return 0;
}
```

```
/*
Output:
97
102.5
*/
```

IF STATEMENT

```
/*
Output:
Comparison correct. x equals 5
The name is Bill. Bill Gates
That is correct 5 is not 6
Yup 10 is greater than 5
*/
```

```
int main(){
    int x = 5;
    string name = "Bill Gates";

    if (x == 5){
        cout << "Comparison correct. x equals 5" << "\n";
    }

    if (name == "Bill Gates"){
        cout << "The name is Bill. Bill Gates" << "\n";
    }

    if (x != 6){
        cout << "That is correct 5 is not 6" << "\n";
    }

    if (10 > x){
        cout << "Yup 10 is greater than 5" << "\n";
    }

    if (10 < x){
        cout << "This line will not execute" << "\n";
    }
    return 0;
}
```

IF STATEMENT

&& And

|| Or

! Not

MORE IF STATEMENTS

```
int main(){
    int number = 5;
    string name = "Bill Gates";

    if ((number >= 5) && (name == "Hello")){
        cout << "First if statement works" << "\n";
    }

    if ((number >= 5) || (name == "Hello")){
        cout << "Second if statement works" << "\n";
    }

    if (!number == 5){
        cout << "Hello" << "\n";
    }

    return 0;
}
```

```
/*
Output:
Second if statement works
*/
```

EVEN MORE IF STATEMENTS

```
If (condition){
        // block of code
)
else if(condition){
        // block of code
}
else{
        // block of code
}
```

```
int status(string name){
    int val;
    if (name == "Bill"){
        val = 1;
    }
    else if (name == "Steve"){
        val = 2;
    }
    else{
        val = 3;
    }

    return val;
}

int main(){
    string name1 = "Bill";
    string name2 = "Steve";
    string name3 = "Alan";

    cout << status(name1) << "\n";  //print 1
    cout << status(name2) << "\n";  //print 2
    cout << status(name3) << "\n";  //print 3
    return 0;
}
```

LOOPS

- Tells the compiler to read over a piece of code a specified number of times
- BEWARE of infinite loops

WHILE && FOR LOOP

```
int main (){
    int num = 0;

    while (num!= 5){
        cout << num << "\n";
        num++;
    }

    return 0;
}
```

```
int main ()
{
    for (int num = 0; num <5; num++){
        cout << num << "\n";
    }
    return 0;
}
```

```
int main() {
    int num = 0;

    do {
        cout << num << "\n";
        num++;
    }while (num != 5);

    return 0;
}
```

```
/*
Output:
0
1
2
3
4
*/
```

ARRAYS

- Sometimes a variable can hold multiple values. This data type is called arrays
- Can arrays be made up of arrays?

STATIC MEMORY ARRAY

```
int main (){
    string name[2] = {"Hi", "Bye"};
    cout << name[0] << "\n";
    cout << name[1] << "\n";

    name[1] = "Hello";
    cout << name[1] << "\n";

    return 0;
}
```

```
/*
Output:
Hi
Bye
Hello
*/
```

ARRAY LOOP

```
int main (){

    string name[] = {"Bill", "Steve", "Charles", "Alan"};

    int length = sizeof(name)/sizeof(name[0]);

    for (int i = 0 ; i < length; i++){
        cout << name[i] << "\n";
    }

    return 0;
}
```

```
/*
Output:
Bill
Steve
Charles
Alan
*/
```

DYNAMIC MEMORY ARRAY

```
int main (){
    vector <string> name;

    for (int i = 0; i <5; i++){
        string value;
        cin >> value;
        name.push_back(value);
    }

    for (int j = 0; j < name.size(); j++){
        cout << name[j] << "\n";
    }

    return 0;
}
```

```
/*
Input:
Hello
my
name
is
Bill

Output:
Hello
my
name
is
Bill
*/
```

STRUCTS

```
int main (){
    struct Employee{
        string name;
        int number;
        double wage;
    };

    Employee bill;
    bill.name = "Bill";
    bill.number = 12345;
    bill.wage = 6140.50;

    cout << bill.name << " " << bill.wage << "\n";

    Employee steve ={"Steve", 54321, 6645.20};
    cout << steve.name << " " << steve.wage << "\n";

    return 0;
}
```

```
/*
Bill 6140.5
Steve 6645.2
*/
```

TEMPLATE

```cpp
#include<bits/stdc++.h>
#define endl "\n"
#define print(x) cout << x << endl

using namespace std;


int main (){

    print ("Hello World");

    return 0;
}
```

KATTIS && GITHUB

- https://open.kattis.com/
  - Hello World! (0.5 point)
  - A Different Problem (1 point)
  - Cold-puter Science (1 points)
  - Bus Numbers (1.5 points)
  - Server (1.5 points)
  - Ptice (2 points)
  - Natrij (2 points)
  - In Or Out (4 points)
  - Path Tracing (4 points)
- https://github.com/