**Week 1 (Introduction to C++) *– Summary***

**Key Topics**

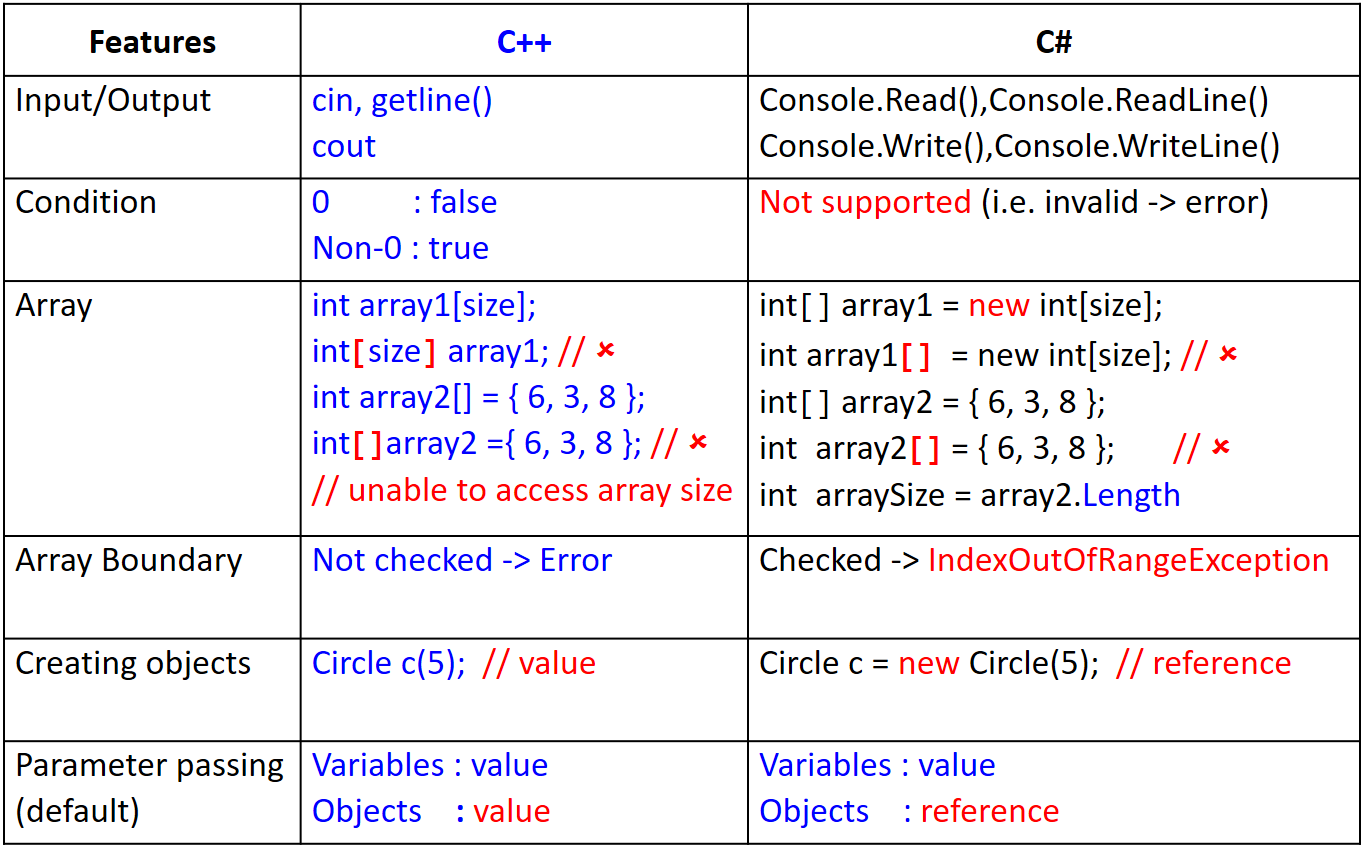
* **C++**
* **Pointers**
* **Class**

**C++**

**Sample Program Structure**

|  |
| --- |
| #include "stdafx.h" // included by Visual Studio automatically  #include <iostream> // for input/output  using namespace std; // for std C++ definitions: cin, cout, string, . . .  int main()  {  cout << "Hello, World!" << endl;  system("PAUSE");  return 0;  } |

**C++ vs C#**

****

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // to flush the input buffer

**Pointers**

**&** -> reference operator (retrieves the address of data variable)

**\*** -> de-reference operator (retrieves the data pointed to by the pointer)

|  |
| --- |
| int main()  {  int x; // x stores data  int **\***y; // y stores address (pointer)  x = 10; // assign 10 to x  y = **&**x; // assign memory address of x to y  cout << x << endl; // display data in x  cout << y << endl; // display data (address) in y  cout << **\***y << endl; // display data pointed to by y  cout << y << endl; // display data in y (address of x)  cout << **&**x << endl; // display address of x  cout << **&**y << endl; // display address of y  // x = y; // error -> x cannot store address  // y = x; // error -> y cannot store data value  } |

Parameter Passing by Value/Reference

|  |
| --- |
| void changeValue1(int x, int **\***y)  {  x = x + 10;  \*y = \*y + 10;  }  void changeValue2(int x, int **&**y)  {  x = x + 10;  y = y + 10;  } |

// calling the function

changeValue1(x, **&**y); // need to pass address

changeValue2(x, y); // no need to pass address

// converted in function itself

**Class** (e.g. Employee class)

**Employee.h** (Specification)

|  |
| --- |
| #pragma once  #include <string>  using namespace std;  class Employee  {  // attributes (properties)  private:  string firstName;  string lastName;  int salary;  // functions (methods)  public:  Employee(); // default constructor  ~Employee(); // default destructor  **Employee(string fn, string ln, int s);** // parameterized constructor  void setFirstName(string fn);  string getFirstName();  void setLastName(string ln);  string getLastName();  void setSalary(int s);  int getSalary();  }; |

**Employee.cpp** (Implementation)

|  |
| --- |
| #include "stdafx.h"  #include "Employee.h"  Employee::Employee() { } // default constructor  Employee::Employee() { } // default destructor  Employee::**Employee(string fn, string ln, int s)** // parameterized constructor  **{**  **firstName = fn;**  **lastName = ln;**  **salary = s;**  **}**  void Employee::setFirstName(string fn) { firstName = fn; }  string Employee::getFirstName() { return firstName; }  void Employee::setLastName(string ln) { lastName = ln; }  string Employee::getLastName() { return lastName; }  void Employee::setSalary(int s) { **salary = (s > 0) ? s : 0;** }  int Employee::getSalary() { return salary; } |

main()

|  |
| --- |
| int main()  {  // create Employee objects  Employee e1("Terri", "Ong", 6000);  Employee e2("Brian", "Lee", 5000);  // display yearly salary of each Employee  cout << e1.getSalary() \* 12 << endl;  cout << e2.getSalary() \* 12 << endl;  // increase salary of each Employee by 10%  e1.setSalary(e1.getSalary() \* 110 / 100);  e2.setSalary(e2.getSalary() \* 110 / 100);  // display yearly salary of each Employee  cout << e1.getSalary() \* 12 << endl;  cout << e2.getSalary() \* 12 << endl;  } |