| Name: | | Course Code: | |
|---|---|---|---|

# Workbook

# Contents

# Course Introduction

Welcome to grade 12 computer science/programming (ICS4U/4C). You will be learning how to program a computer using the C# programming language.

## Mark Breakdown

| Category | Percent |
|---|---|
| **Term Work**<br>C# Workbook, Coding Challenges, Written Tests | 60% |
| **Senior Project**<br>Write a proposal document (if you didn't already write it in grade 10) and start development on a project. It can be a game, mobile app, web application, robotics application, etc. | 20% |
| **Exam** | 20% |

## Email

The majority of communication between you and I will be via your school email account. Make sure you check your school email account at the start and at the end of each period.

## GitHub

You will be using an industry-standard code storage and versioning system called GitHub in this course. All your work will be pushed (uploaded) to GitHub daily, so I have constant access to your work. When you want me to mark something, you simply must make sure your work is pushed to GitHub and that you send me a notification email.

I have created a private repository (storage area) on GitHub for you. To access it, you need to have a GitHub account. So, head over to https://github.com/ and create an account. Then email me your GitHub username so I can add you as a collaborator on the repository.

NOTE: You can access GitHub from school and from home.

Follow along with https://youtu.be/hMymT4qJB2c to setup and use GitHub.

## Course Software

All the software in this course is free to download and use. This means you can set up your home computer just like your one at school.

## Daily Workflow

Follow this workflow whenever you work on this course (at school or at home):

1. Log into your computer.
2. Pull your repository from GitHub.
3. Check email.
4. Work on this workbook, do coding challenges, work on final project, etc.
5. Commit and push your repository to GitHub.
6. Send me any required notification emails.
7. Log off your computer.

## Mastery System

The best way to learn to program is to keep working away at it until you've mastered the basics. That's why I require you to keep redoing your assignments in this course until they are perfect. I mark everything on a 4-point scale:

| **Level** | **Description** |
|---|---|
| 1 | You are just beginning to understand a concept. |
| 2 | Your understanding is developing. |
| 3 | You are proficient, but still make some mistakes. |
| 4 | You have mastered a concept and very rarely make mistakes. |

All lessons in this workbook must be completed to LEVEL 4 before you can go onto the course Coding Challenges. Any given Coding Challenge must be mastered to LEVEL 4 before you can write the Coding Test associated with that Coding Challenge.

## C++ Workbook

You will learn the basics of the C++ programming language by completing the lessons in this workbook. When you have completed a lesson in this workbook, ensure it is pushed to GitHub and then send me a notification email.

**You must complete all lessons in this workbook to LEVEL 4 before going on to the course Coding Challenges.**

## Coding Challenges

There are several Coding Challenges in the course. This is where you really learn to solve problems and program! Each challenge is followed by a written test which tests the concepts learned during the challenge.

**You must achieve LEVEL 4 on a Coding Challenge before you can attempt the test.**

## Coding Tests

After completing a Coding Challenge to LEVEL 4, you are eligible to write the Coding Test associated with the challenge. The test will cover the basic principles learned in the Coding Challenge.

**You are <u>encouraged</u> (but not required) to write the Coding Tests over and over until you reach LEVEL 4.**

## Senior Project

You will be working on a senior project in this course, which will span grades 11 and 12. It can be a game, mobile app, web application, desktop application, robotics application, or anything you can think of. This project will be worked on in grades 11 and 12. If you did not submit a project proposal in grade 10 or 11, then you will start with that.

## Final Exam

The final exam is similar to the Coding Tests and written on paper.

# Setting Up Your IDE

You can use Code::Blocks or Visual Studio Community to program in C++. I prefer Visual Studio, but Code::Blocks is good to.

Code::Blocks: http://www.codeblocks.org/downloads

Visual Studio Community: https://visualstudio.microsoft.com/downloads/

Setting up a project in CodeBlocks: https://goo.gl/AvmRRw

Setting up a project in Visual Studio Community: https://goo.gl/dNK9uy

# SoloLearn – Basic Concepts – What is C++

Head over to https://www.sololearn.com/ create an account and start the C++ course. Complete the ***What is C++?*** lesson and then answer the following questions.

1. What is C++?

2. List *three* applications of C++.

# SoloLearn – Basic Concepts – Hello, world!

Complete the ***Hello, world!*** lesson and then answer the following questions

1. Answer the following questions based on the simple program below.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "Hello, world!";
7      return 0;
8  }
```

a. What is the purpose the #include statement on line 1?

b. Why is the std namespace specified on line 2?

c. What is special about the main function defined on line 4?

d. What is the purpose of the cout statement on line 6?

e. What is the purpose of the return statement on line 7?

# SoloLearn – Basic Concepts – Getting the Tools

Complete the ***Getting the Tools*** lesson and then answer the following questions.

1. What is an IDE?

2. What is a compiler?

# SoloLearn – Basic Concepts – Printing a Text

Complete the ***Printing a Text*** lesson and then answer the following questions.

1. What is the purpose of the *endl* manipulator?

2. What does the \n escape character do when inserted into a string?

# SoloLearn – Basic Concepts – Comments

Complete the ***Comments*** lesson and then answer the following questions.

1. What are the two methods of adding comments to your code?

# SoloLearn – Basic Concepts – Variables

Complete the ***Variables*** lesson and then answer the following questions.

1. What is a ***variable***?

2. What is an ***identifier***?

3. What are the rules for identifiers?

# SoloLearn – Basic Concepts – Working with Variables

Complete the ***Working with Variables*** lesson and then answer the following questions.

1. Write a program which calculates the length of the hypotenuse of a triangle given the length of the other two sides. The program should prompt the user for the two side lengths and then calculate the length of the hypotenuse. You will want to use the *cmath* header file for the *sqrt* function.

# SoloLearn – Basic Concepts – Basic Arithmetic

Complete the ***Basic Arithmetic*** lesson and then answer the following questions.

1. Predict the output of the following program.

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

int main()
{
    cout << (10 / (3 + 12) + 20) % 3 << endl;
    cout << (3 + 6 - 2 / 3 * (10 - 7 % 2)) << endl;
    cout << (15 % 16 / 2 + 10) - 20 << endl;
    return 0;
}
```

# SoloLearn – Basic Concepts – Assignment and Increment Operators

Complete the ***Assignment and Increment Operators*** lesson and then answer the following questions.

1. State the difference between the prefix and postfix increment operators.

2. Write three different ways of incrementing the variable *x* by one.

# SoloLearn – Conditionals and Loops – The if Statement

Complete ***The if Statement*** lesson and then answer the following questions.

1. Write a program which prompts the user for an integer and then informs the user if the number is greater than 10.

# SoloLearn – Conditionals and Loops – The else Statement

Complete ***The else Statement*** lesson and then answer the following questions.

1. Write a program which prompts the user for two integers. It then informs the user which number is greater or if they are equal.

# SoloLearn – Conditionals and Loops – The while Loop

Complete ***The while Loop*** lesson and then answer the following questions.

1. Write a program which asks the user for a positive number and then calculates the sum of the integers from 1 up to and including the number entered.

# SoloLearn – Conditionals and Loops – Using a while Loop

Complete the ***Using a while Loop*** lesson and then answer the following questions.

1. Calculate and output the Fibonacci Sequence using a while loop (no recursion). Prompt the user for the number of terms to display. The program should repeat until the user enters 0 for the number of terms. Your output should match https://youtu.be/vu2Sl2ohjus

# SoloLearn – Conditionals and Loops – The for Loop

Complete ***The for Loop*** lesson and then answer the following questions.

1. Predict the output of the following program.

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

int main()
{
    for (int x = 0; x < 6; x += 3)
    {
        for (int y = 0; y < 8; y += 2)
        {
            int z = (x * y / 2 + 10) % 8;
            cout << z << " ";
        }
    }

    return 0;
}
```

2. Refactor your Fibonacci Calculator (from the last lesson) to use for loops instead of while loops.

# SoloLearn – Conditionals and Loops – The do…while Loop

Complete ***The do…while Loop*** lesson and then answer the following questions.

1. What is the difference between the *while loop* and the *do-while loop*?

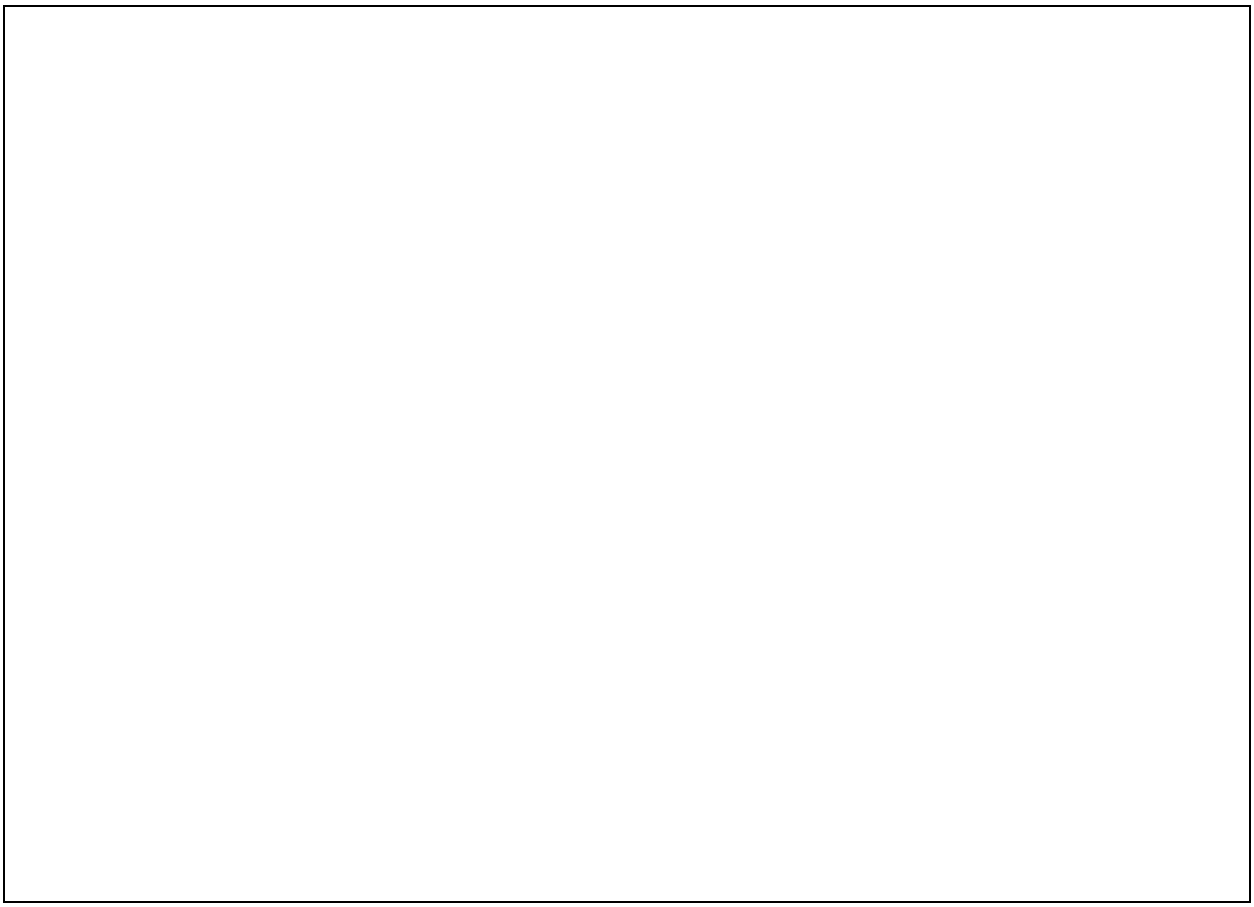
2. Predict the output of the following program.

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

int main()
{
    int x = 42;
    do
    {
        int y = 0;
        while (y++ < 4)
        {
            for (int i = 3; i > 0; i--)
            {
                cout << (x - y) % i << endl;
            }
        }
    } while (x < 42);

    return 0;
}
```

3. Refactor your Fibonacci Calculator to use do-while loops.



# SoloLearn – Conditionals and Loops – The switch Statement

Complete ***The switch Statement*** lesson and then answer the following questions.

1. Why might you choose to use a switch statement instead of an if statement?

2. What is the purpose of the default case?

# SoloLearn – Conditionals and Loops – Logical Operators

Complete the ***Logical Operators*** lesson and then answer the following questions.

1. Predict the output of the following program.

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

int main()
{
    int x = 10;
    int y = 20;
    int z = 30;

    cout << (x < y && z > y) << endl;
    cout << (z != (x + y) || z - y <= x) << endl;
    cout << !(z - (x + y)) << endl;
    cout << ((x + y) / z > 0 && z % y > x) << endl;

    return 0;
}
```

# SoloLearn – Data Types, Arrays, Pointers – Introduction to Data Types

Complete the ***Introduction to Data Types*** lesson and then answer the following questions.

1. What is a ***data type***?

2. Describe each of the following data types:  integer, float, character, string, Boolean.

# SoloLearn – Data Types, Arrays, Pointers – int, float, double

Complete the ***int, float, double*** lesson and then answer the following questions.

1. What is the purpose of the *signed* and *unsigned* keywords?

2. What is the purpose of the *short* and *long* keywords?

3. Describe the differences between the *three* types of floating point data types: *float, double,* and *long double***.**

4. The following code gives the output https://youtu.be/mHK1haqXlPM. Explain why the while loop terminates.

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

int main()
{
    // Wait until the user presses the ENTER key before
    // continuing.
    cin.ignore();

    signed short int x = 1;
    while (x > 0) cout << ++x << endl;
    return 0;
}
```

# SoloLearn – DataTypes, Arrays, Pointers – string, char, bool

Complete the ***string, char, bool*** lesson and then answer the following questions.

1. What header must be included if you want to use the string data type, assuming you haven't included <iostream>.

2. Predict the output of the following program.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       bool a = true;
7       bool b = false;
8
9       cout << ((a && b) || (a || !b)) << endl;
10      cout << (!(a && b) && !(a || b)) << endl;
11      cout << (a && !(a || b) && !a) << endl;
12  }
```

# SoloLearn – Data Types, Arrays, Pointers – Variable Naming Rules

Complete the ***Variable Naming Rules*** lesson and then answer the following questions.

1. State the variable naming rules.

2. What are the *two* main naming conventions when programming in C++? Give examples.

3. C++ is case-sensitive. What does this mean?

# SoloLearn – Data Types, Array, Pointers – Arrays

Complete the ***Arrays*** lesson and then answer the following questions.

1. Predict the output of the following program.

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

int main()
{
    int a[] = { 5, 3, 1, 4, 0 };

    cout << a[0] << endl;
    cout << a[1] << endl;
    cout << a[4] << endl;
    cout << a[5] << endl;
}
```

# SoloLearn – Data Types, Arrays, Pointers – Using Arrays in Loops

Complete the ***Using Arrays in Loops*** lesson and then answer the following questions.

1. Write a program [ https://youtu.be/OzIYf7OYxaY ] which prompts the user to enter two integer arrays, each with 5 elements. It then adds the arrays together.

# SoloLearn – Data Types, Arrays, Pointers – Arrays in Calculations

Complete the ***Arrays in Calculations*** lesson and then answer the following questions.

1. Will the following program crash? Explain.

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

int main()
{
    int arr[] = {11, 35, 62, 555, 989};
    int sum = 0;

    for (int x = 0; x < 10; x++) {
        sum += arr[x];
    }

    cout << sum << endl;

    return 0;
}
```

# SoloLearn – Data Types, Arrays, Pointers – Multi-Dimensional Arrays

Complete the ***Multi-Dimensional Arrays*** lesson and then answer the following questions.

1. Predict the output of the following program.

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

int main()
{
    int a[4][4] = {
        { 1, 2, 3, 4},
        { 5, 6, 7, 8},
        { 9, 10, 11, 12},
        { 13, 14, 15, 16}
    };

    for (int i = 0; i < 4; i++)
    {
        for (int j = 3; j >= 0; j--)
        {
            cout << a[i][j] << endl;
        }
    }

    return 0;
}
```

# SoloLearn – Data Types, Arrays, Pointers – Introduction to Pointers

Complete the ***Introduction to Pointers*** lesson and then answer the following questions.

1. How can the memory address of a variable be accessed?

2. What is a pointer?

3. Why are pointers important in C++?

4. What is the data type of all pointers?

5. How is a pointer declared?

6. Determine the output of the following program.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int x = 42;
7       int y = 10;
8       int *xPtr = &x;
9
10      cout << *xPtr + y << endl;
11
12      return 0;
13  }
```

# SoloLearn – Data Types, Arrays, Pointers – More on Pointers

Complete the ***More on Pointers*** lesson and then answer the following questions.

1. What does the *dereference operator (*)* do?

# SoloLearn – Data Types, Arrays, Pointers – Dynamic Memory

Complete the ***Dynamic Memory*** lesson and then answer the following questions.

1. What is the *stack*?

2. What is the *heap*?

3. What does the *new* keyword do?

4. In the following code, in which memory (stack or heap) is the value of x stored? In which memory is the pointer stored? Also, predict the output of the program.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int *x = new int;
7       *x = 2;
8       cout << *x * *x << endl;
9       delete x;
10      return 0;
11  }
```

5. What is a memory leak and how are they caused?

6. What is a *dangling pointer*?

7. What is a *null pointer*?

8. When is *dynamic memory allocation* useful?

9. Write a program [ https://youtu.be/fLo-b19N404 ] which adds two arrays together to form a new array. The program should ask the user the array size and then dynamically allocate heap memory to store the array. Watch out for memory leaks!

# SoloLearn – Data Types, Arrays, Pointers – The sizeof() Operator

Complete ***The sizeof() Operator*** lesson and then answer the following questions.

1. What is the *sizeof()* operator used for?

2. On a particular system, a long int is 4 bytes. On the same system how much memory would a 200 element array of long ints use?

# SoloLearn – Functions – Introduction to Functions

Complete the ***Introduction to Functions*** lesson and then answer the following questions.

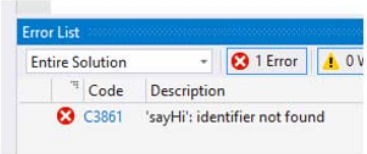
1. What is a *function*?

2. List *four* advantages of using functions.

3. What is the return type of a function which doesn't return a value?

4. The following program does not compile. Fix it without changing the order of the functions.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       sayHi();
7       return 0;
8   }
9
10  void sayHi()
11  {
12      cout << "Hi!" << endl;
13  }
```



# SoloLearn – Functions – Function Parameters

Complete the ***Function Parameters*** lesson and then answer the following questions.

1. Predict the output of the following program. Explain you answer.

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

void cube(int);

int main()
{
    int x = 3;
    cube(x);
    cout << x << endl;
    return 0;
}

void cube(int x)
{
    x = x * x * x;
}
```

# SoloLearn – Functions – Functions with Multiple Parameters

Complete the ***Functions with Multiple Parameters*** lesson and then answer the following questions.

1. Write a function which accepts two strings, concatenates them into a new string, and then returns it. Call your function from the main function and cout the concatenated string.

# SoloLearn – Functions – The rand() Function

Complete ***The rand() Function*** and then answer the following questions.

1. Which header must be included before using *rand()*?

2. What is the purpose of the *srand()* function?

3. What is the purpose of the *time()* function? Which header must be included to use it?

4. Write a function which simulates rolling a multisided die. The function should accept the number of sides and return the random roll. Ensure that your die rolls are as random as possible. Here is the signature.

```
int roll(int numSides)
```

# SoloLearn – Functions – Default Arguments

Complete the ***Default Arguments*** lesson and then answer the following questions.

1. When might a programmer want to use default function arguments?

2. Predict the output of the following program.

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

int calc(int a = 1, int b = 2, int c = 3)
{
    return (a + b) * c % b;
}

int main()
{
    int a = 5;
    int b = 4;
    int c = 0;

    while (a + --b > ++c)
    {
        cout << calc(a, c) << endl;
    }

    return 0;
}
```

# SoloLearn – Functions – Function Overloading

Complete the ***Function Overloading*** lesson and then answer the following questions.

1. Why might a programmer want to overload a function?

2. Predict the output of the following program.

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

int sum(int, int);
double sum(double, double);
string sum(string, string);

int main()
{
    cout << sum(1, 2) << endl;
    cout << sum(1.23, 4.56) << endl;
    cout << sum("Hello, ", "world!") << endl;

    return 0;
}

int sum(int a, int b)
{
    return a + b;
}

double sum(double a, double b)
{
    return a + b;
}

string sum(string a, string b)
{
    return a + b;
}
```

# SoloLearn – Functions – Recursion

Complete the ***Recursion*** lesson and then answer the following questions.

1. What is a *recursive function*?

2. What is a *termination condition*?

3. Rewrite your Fibonacci Calculator to use recursion instead of iteration.

# SoloLearn – Functions – Passing Arrays to Functions

Complete the ***Passing Arrays to Functions*** lesson and then answer the following questions.

1. Write a function which accepts an array of floats and the length of the array and returns the average of the floats. Here is the signature.

```
float average(int n, float arr[])
```

# SoloLearn – Functrions – Pass by Reference with Pointers

Complete the ***Pass by Reference with Pointers*** lesson and then answer the following questions.

1. What does it mean to pass an argument by *value*?

2. What does it mean to pass an argument by *reference*?

3. How does C++ pass arguments by default?

4. Predict the output of the following program. Explain your answer.

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

void calc(int);
void calc(int*);

int main()
{
    int x = 2;
    calc(x);
    cout << x << endl;
    calc(&x);
    cout << x << endl;
    return 0;
}

void calc(int x)
{
    x *= 2;
}

void calc(int *x)
{
    *x *= 2;
}
```

5. Write a function that accepts two integer arrays and their sizes and returns a pointer to a new array containing all the elements. Here is the signature.

```
int* merge(int a[], int na, int b[], int nb)
```

# SoloLearn – Classes and Objects – What is an Object

Complete the ***What is an Object*** lesson and then answer the following questions.

1. What is the goal of *Object Oriented Programming (OOP)*?

2. What is an *object*?

3. What is an *attribute*?

4. What is a *behaviour*?

# SoloLearn – Classes and Objects – What is a Class

Complete the ***What is a Class*** lesson and then answer the following questions.

1. What is a *class*? What is the relationship between classes and objects?

2. What is a *property*?

3. What is a *method*?

4. Describe an Enemy class for a video game. The class should have *three* properties and *three* methods.

# SoloLearn – Classes and Objects – Example of a Class

Complete the ***Example of a Class*** lesson and then answer the following questions.

1. What is an *access specifier*?

2. What is the purpose of the *public* access specifier?

# SoloLearn – Classes and Objects – Abstraction

Complete the ***Abstraction*** lesson and then answer the following questions.

1. What is *data abstraction*?

2. Why is abstraction important for OOP?

# SoloLearn – Classes and Objects – Encapsulation

Complete the ***Encapsulation*** lesson and then answer the following questions.

1. What is *encapsulation*?

2. What is *data hiding*? Why would a programmer want to use data hiding?

3. What is *black boxing*?

4. List *three* benefits of encapsulation.

# SoloLearn – Classes and Objects – Example of Encapsulation

Complete the ***Example of Encapsulation*** lesson and then answer the following questions.

1. Implement the Enemy class you described earlier. The class should contain public and private members, *three* properties, and *three* methods.

# SoloLearn – Classes and Objects – Constructors

Complete the ***Constructors*** lesson and then answer the following questions.

1. What is a *constructor*? Why are they useful?

2. What is the return type of a constructor?

3. Is it possible to overload a constructor?

4. Create a constructor for your Enemy class.

# SoloLearn – More On Classes – Separate Files for Classes

Complete the ***Separate Files for Classes*** lesson and then answer the following questions.

1. What is the purpose of the header file (.h)?

2. What is the purpose of the source file (.cpp)?

3. What is the *scope resolution operator (::)* used for?

4. Put your Enemy class in its own .cpp file with a .h file containing function prototypes and variable declarations.

Paste the .h file here…

Paste the .cpp file here…

# SoloLearn – More On Classes – Destructors

Complete the ***Destructors*** lesson and then answer the following questions.

1. What is a ***destructor***?

2. When is an object's destructor destroyed?

3. When are destructors useful?

4. Can a destructor be overloaded? Explain.

5. Add a destructor to your Enemy class. Be sure to declare the destructor in the header file.

Paste the .h file here…

Paste the .cpp file here…

# SoloLearn – More On Classes – Selection Operator

Complete the ***Selection Operator*** lesson and then answer the following questions.

1. What is the purpose of *#ifndef*?

2. What is the purpose of the *dot operator (.)?*

3. How is the *selection operator (->)* similar to the *dot operator (.)*? How is it different?

4. Instantiate your Enemy class, create a pointer to the Enemy object, and then call the Attack method using the selection operator.

# SoloLearn – More On Classes – Const Objects

Complete the ***Const Objects*** lesson and then answer the following questions.

1. What is the ***const*** keyword used for?

2. Why might a programmer want to use a constant object?

# SoloLearn – More On Classes – Member Initializers

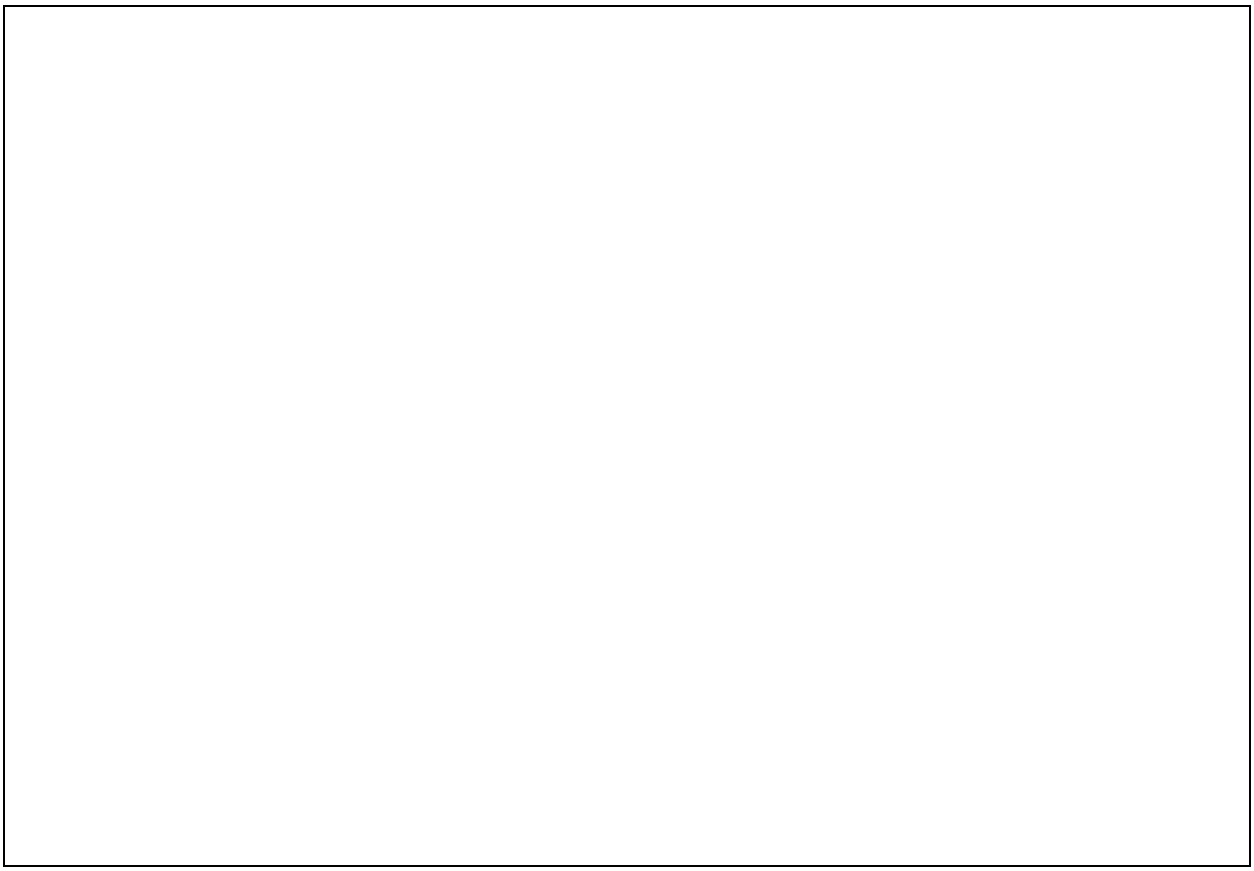
Complete the ***Member Initializers*** lesson and then answer the following questions.

1. When should a *member initialization list* be used?

2. Add a constant member (called *id*) to your Enemy class and use a member initialization list in the constructor.

.h file…

.cpp file



# SoloLearn – More On Classes – Composition

Complete the ***Composition*** lessons and then answer the following questions.

1. What is *composition*?

2. When might a programmer decide to use composition?

# SoloLearn – More On Classes – The Friend Keyword

Complete ***The Friend Keyword*** lesson and then answer the following questions.

1. Why might a programmer want to use *friend functions*?

# SoloLearn – More On Classes – The This Keyword

Complete ***The This Keyword*** lesson and then answer the following questions.

1. What is the *this* keyword used for?

# SoloLearn – More On Classes – Operator Overloading

Complete the ***Operator Overloading*** lesson and then answer the following questions.

1. Why might a programmer want to overload an operator?

# SoloLearn – Inheritance & Polymorphism – Inheritance

Complete the ***Inheritance*** lesson and then answer the following questions.

1. What is *inheritance*?

2. Create a daughter class of your Enemy class.

# SoloLearn – Inheritance & Polymorphism – Protected Members

Complete the ***Protected Members*** lesson and then answer the following questions.

1. How is the *protected* access specifier similar to the *private* access specifier? How is it different?

2. Describe the *three* types of inheritance: *public*, *protected*, and *private*.

# SoloLearn – Inheritance and Polymorphism – Derived Class Constructor & Destructor

Complete the ***Derived Class Constructor & Destructor*** lesson and then answer the following questions.

1. Are the ctor and dtor of a base class called when a derived class is instantiated? Explain.

2. When a derived class is instantiated, which ctor is called first: the base class ctor or the derived class ctor? Which dtor is called first?

# SoloLearn – Inheritance and Polymorphism – Polymorphism

Complete the ***Polymorphism*** lesson and then answer the following questions.

1. What does the term *polymorphism* mean?

2. Why would a programmer want to use *polymorphism*?

# SoloLearn – Inheritance and Polymorphism – Virtual Functions

Complete the ***Virtual Functions*** lesson and then answer the following questions.

1. What are *virtual functions* used for?

2. What is a *polymorphic class*?

# SoloLearn – Inheritance and Polymorphism – Abstract Classes

Complete the ***Abstract Classes*** lesson and then answer the following questions.

1. Why might a programmer use a *pure virtual function*?

2. What happens if a derived class fails to override a pure virtual function in the base class?

3. What happens if you try to instantiate a class which has a pure virtual function (aka *abstract class*)?

# SoloLearn – Templates, Exceptions, and Files – Function Templates

Complete the ***Function Templates*** lesson and then answer the following questions.

1. Why might a programmer want to use a *function template*?

2. Write a *function template* to sum the contents of an array. Here is the method signature.

```
// a[] is the array, n is the array length
template <class T>
T sum(T a[], int n)
```

# SoloLearn – Templates, Exceptions, and Files – Class Templates

Complete the ***Class Templates*** lesson and then answer the following questions.

1. Why might a programmer want to use a *class template*?

# SoloLearn – Templates, Exceptions, and Files – Template Specialization

Complete the ***Template Specialization*** lesson and then answer the following questions.

1. When might a programmer want to use *template specialization*?

# SoloLearn – Templates, Exceptions, and Files – Exceptions

Complete the ***Exceptions*** lesson and then answer the following questions.

1. What is an *exception*?

2. Why would a programmer want to throw an exception and crash their own program?

3. What is the purpose of the *try/catch* block?

# SoloLearn – Templates, Exceptions, and Files – More on Exceptions

Complete the ***More on Exceptions*** lesson and then answer the following questions.

4. Predict the output of the following program.

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

int main()
{
    int x = 0;
    int y = 5;
    int z = 0;

    while (x < 10)
    {
        try
        {
            if (y == 0)
            {
                throw "Division by zero!";
            }
            else
            {
                z = x / y;
            }
            cout << z << endl;
        }
        catch (...)
        {
            cout << "Division by zero!" << endl;
        }
        x++;
        y--;
    }

    return 0;
}
```

# SoloLearn – Templates, Exceptions, and Files – Working with Files

Complete the ***Working with Files*** lesson and then answer the following questions.

1. Describe the following data types found in the *fstream* library: *ofstream, ifstream,* and *fstream*.

2. Which header files must be included in order to perform file processing?

# Course Coding Assignments

The coding assignments are available in the course Google Drive.  Remember, each assignment has a follow-up test, and you can only write the test after you have achieved **LEVEL 4** on the assignment.

Coding Assignments: https://goo.gl/Ag7dyY

# Senior Project

Now that you have gained some serious programming skill, it's time to think about what you really want to make. Is it a game? A mobile app? A website? A robotics application? The Senior Project is your chance to work on your first big project as a developer.

If you have not already done so, first you need to write up a Software Design Document (SDD). The SDD is a description of how you are going to solve a problem. It outlines the features of a piece of software and serves as a guide while you develop the software. Basically, it the most useful tool for making sure the right work gets done on a project.

Check the course Google Drive for the SDD templates [ https://goo.gl/vuks7E ]. If you don't find a SDD that suits your needs, then send me an email and I can make up a custom one for you. There are SDDs for each of the main project types.

Once you have your SDD completed its time to start designing and developing your product! I'm here to help you out so don't be afraid to ask questions!