**THE GAME OF LIFE – CHOOSE!**

Report by

Charottama Oshmar

200158545

**Preface**

Firstly, I would like to thank my parents for everything they have given me. Secondly I would like to thank my lecturers at Binus International University, especially Mr. Bagus Kerthyayana, for assisting and giving consultations to help the completion of this final project of the semester. Lastly I would like to thank my friends, with a special mention to Hansvin, Yosua, Archel, and Aldi, for giving me the motivation and sharing their knowledge with me.

This report was written to describe the final project created to acquire the mark for the Introduction to Programming subject of Binus International University. The theme of the program chosen is a text-based adventure game, based on the *Goosebumps* novel series by R. L. Stine. The game, much like the novels, requires the player to make decisions which would determine the next path they would travel.

This whole final project has helped me to become more adept and has given me the experience to write a program on my own. It took approximately one month to complete the program and this report, and I hope you find my game able to help understand an example of implementation of software engineering.

**Description**

//Title of Project : THE GAME OF LIFE – CHOOSE!

//Name: : Charottama Oshmar

//Student ID : 2001585455

//Purpose : To learn about programming in C++ language in a fun way while also entertaining the user

Apart from being a component of scoring of the syllabus, this program is also created for the general people who do not understand much about programming. This is the reason why the program was created in this fashion, so as to present the user with a sense of entertainment and curiosity. This way, the hope is to intrigue the player and rid the common stereotype that programming is boring and not fun.

Main: Most of the program will run through the main function

At the end of the first part of the story (the first cout) the program will show two options to the user: option A and option B.

Option A: The random number function will be the basis of this option. The randomly-generated number will be put into basic arithmetical operations which the user will have to solve. Clearing the question will face the user with three options:

1A will ask the user to sort an array of numbers

1B will ask the user to add increasing numbers (n-(n-1) + n-(n-2) … + n

1C will ask the user to **win** a game of rock paper scissors

Option B: The program will print (cout) a riddle, which the user must solve. The “if” and “else” function will be the main function for this option. Solving the riddle will face the user with three options:

1A will ask the user to sort an array of numbers

1B will ask the user to multiply increasing numbers (n-(n-1) \* n-(n-2) … \* n

1C will ask the user to **lose** to a game of rock-paper-scissors

Whichever option is chosen the program will end after the second fork.

**Designing the Program:**

**Planning**

**Int main**

Story begins (cout narration)

Option B

Option A

or

Option B

**Option A**

Using the random number

put into basic arithmetic

**Option B**

Using the if and else

in the form of a riddle

If stillAlive == true

Story continues

If stillAlive == true

Story continues

Ending E

Ending C

Ending G

Option H

Losing Rock-Paper-Scissors

Ending H

Ending F

Ending D

Option C

Sorting Descending Array

Option E

Winning Rock-Paper-Scissors

Option G

Multiplying numbers

Option D

Add sum of nums

Option F

Sorting Ascending Array

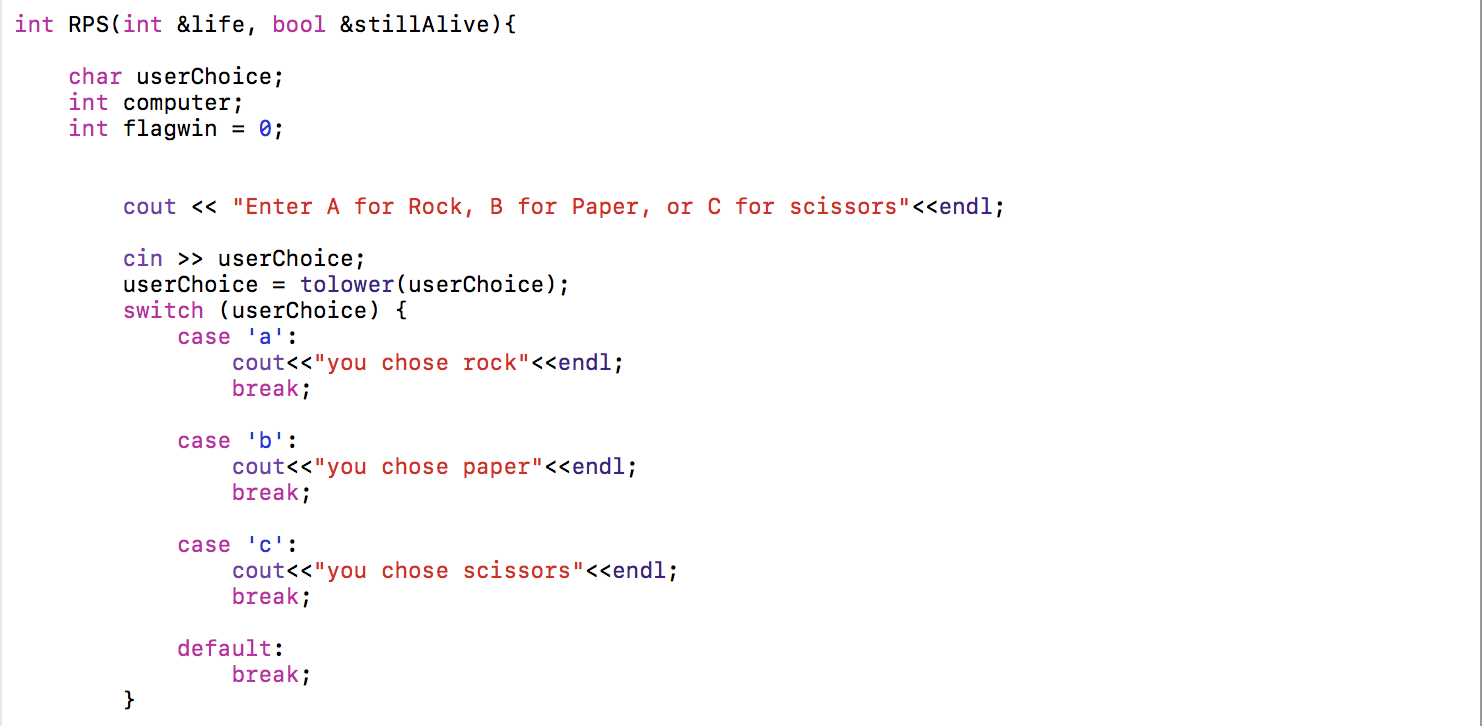
Shown above is the initial schematics of the Game of Life – Choose! As mentioned before, most of the program will run through the main function. The schematic shows the outline of the game, where after clearing a stage of the game and not dying, the player progresses further. There are six different endings. What ending the player gets depends solely on the choices made.

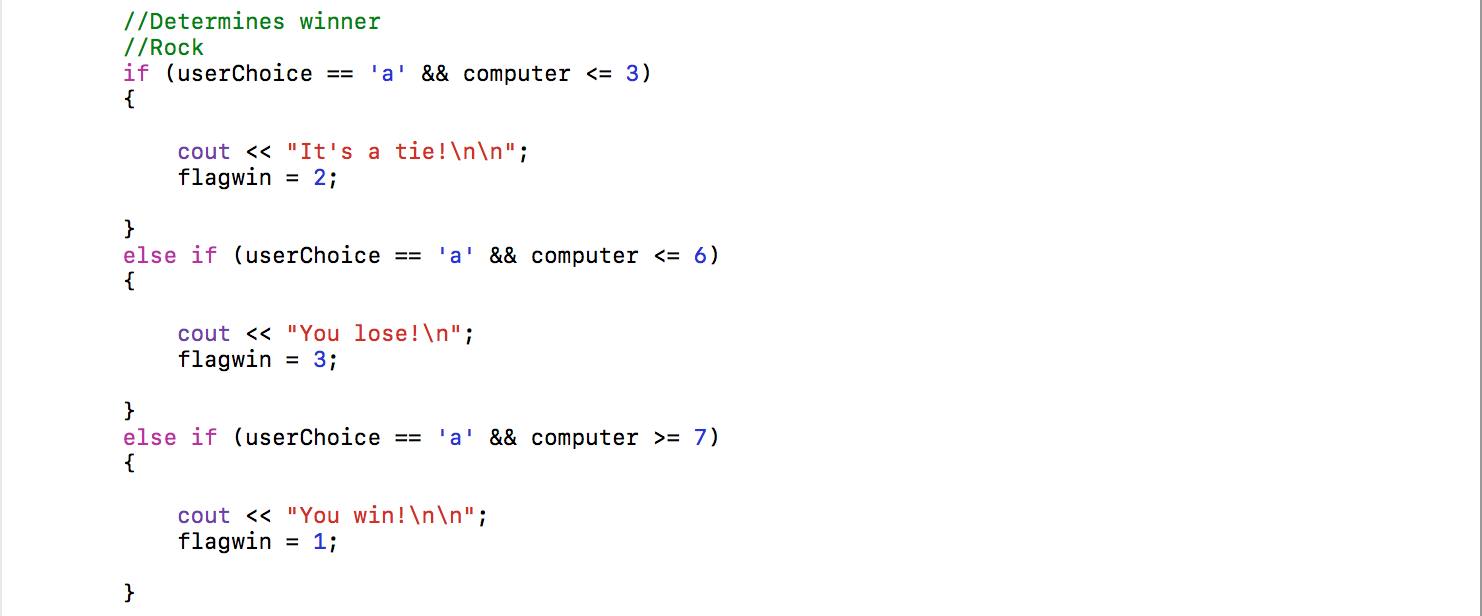
Options A, C, and F use the *seed* and *srand* functions provided in the library of <ctime> in C++, which serves the purpose of generating random numbers. Options D and G employ a class. Options E and H utilize the same function and also the random number generator (RNG).

**Designing the Program:**

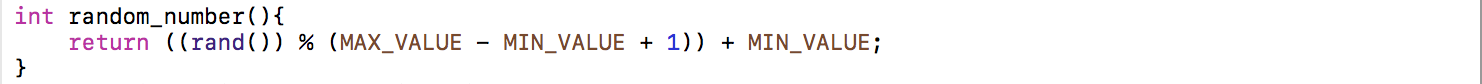
**Functions and Classes**

There are two functions and one class in this program. The first function returns an integer datatype and it is called the *RPS* function (RPS stands for Rock-Paper-Scissors). This function is called twice, in options E and H. The use of the *int* datatype, instead of the *bool* datatype is because there are three possible outcomes, while a Boolean can only hold two possible values. Also, *RPS* applies the *switch* statement, *tolower* function, and makes use of RNG. The result returned is whether the user loses, wins, or ties versus the computer. What the computer chooses will also be determined inside this function. This is what the RNG is for: to assign a scoped number. The choice of the computer will be depending on the random number assigned. From there, the computer can know the winner. Below is a segment of the *RPS* function.

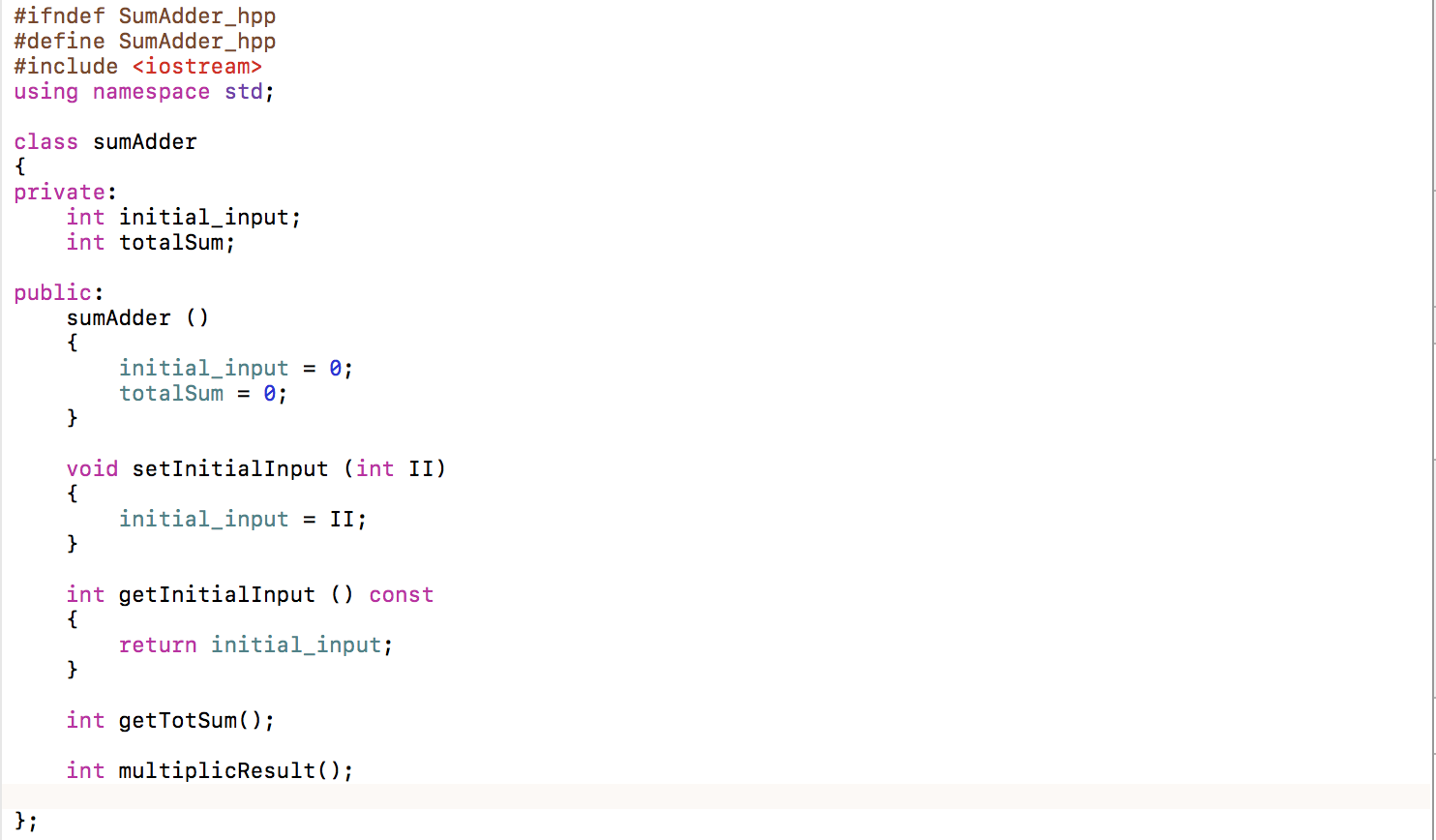




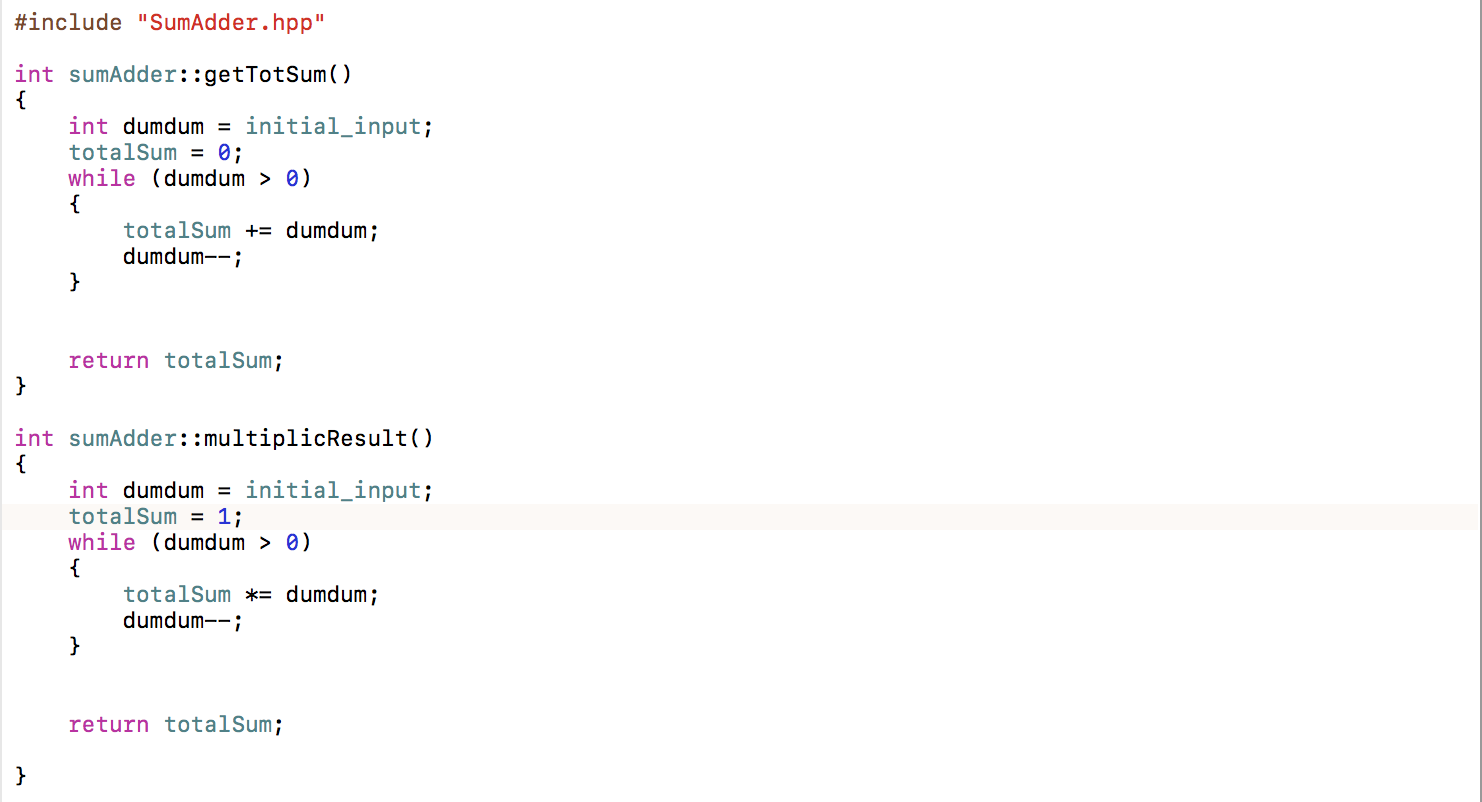
The second function is *random\_number* function. The datatype it returns is an integer. Unlike the *RPS* function, this function is declared above the main function, so it does not need a function prototype. This function is called five times throughout the program. Below is the *random\_number* function.



Finally, there is one class in the game, called the *sumAdder.* This class can add integers and also multiply. The class is called twice, in options E and H. As an addition, this class is separated from the main file and is included in the header file. The class is declared in *SumAdder.hpp,* while the calculation for adding and multiplying the numbers is in *SumAdder.cpp.*



This is the *sumAdder* class, located in *SumAdder.hpp*



These are the methods of the *sumAdder* class, located in *SumAdder.cpp.* The upper part is to calculate additions, while the lower part is used to calculate multiplication.

**Learning:**

**Everything learned**

During the course of this semester, Introduction to Programming has very briefly discussed the hardware of computers. The rest was getting to know a programming language called C++ or CPP.

I believe the reason we learned C++ in the first semester is because it is of moderate difficulty. The materials from the first semester, which is mostly comprised of programming in C++ language would also serve as the basis of the study in the next semester: programming in Java language.

The very basics of C++ include the parts of C++ program, the most important of which is the *int main*, which is the main function, whilst *#include* directives will be read by the compiler before everything else. The *#include* directives will include libraries from the C++ language, and can also include other files into the file being worked on. The language has several basic datatypes, which include, but not limited to, *int* to hold integer numbers, *char* to hold just one character or digit, *string* to hold a series of characters and digits, and *bool* (Boolean) which is a binary variable to hold “true” or “false” expressions. Displaying something on the screen uses the *cout* command while the *cin* command is to contain something inputted by the user.

For the C++ program to determine if a condition is true or false, it needs to be able to make a decision. This is where the *if else if* statement comes in. As the name suggests, the statement would execute a reaction if said condition were true, but if not, the program would branch to a different reaction instead.

Looping is a major part in almost every program, not just in C++, because some parts of it need to be executed more than once. In C++, looping is divided into two groups: pretest and posttest. Pretest loops have two kinds, which are the *for* and *while* loops. Pretest means it tests its statement before the loop begins. Posttest on the other hand is the opposite. It will always loop, even if only once and the condition is wrong to begin with, and only then will it check if the condition is true or false. The posttest only has one member loop called the *do-while.*

Functions, just like the *int main* or main function, can contain lines of program that serve to accomplish a specific task. The use of this is to segregate unique objectives, or purposes, from the main function so that it can be easier to read and also debug when a problem arises. Arrays are rows of numbers, which, among others, can be displayed, sorted, and even pointed at using a *pointer.* A *pointer* can be used to point the address of the object pointed or the object itself.

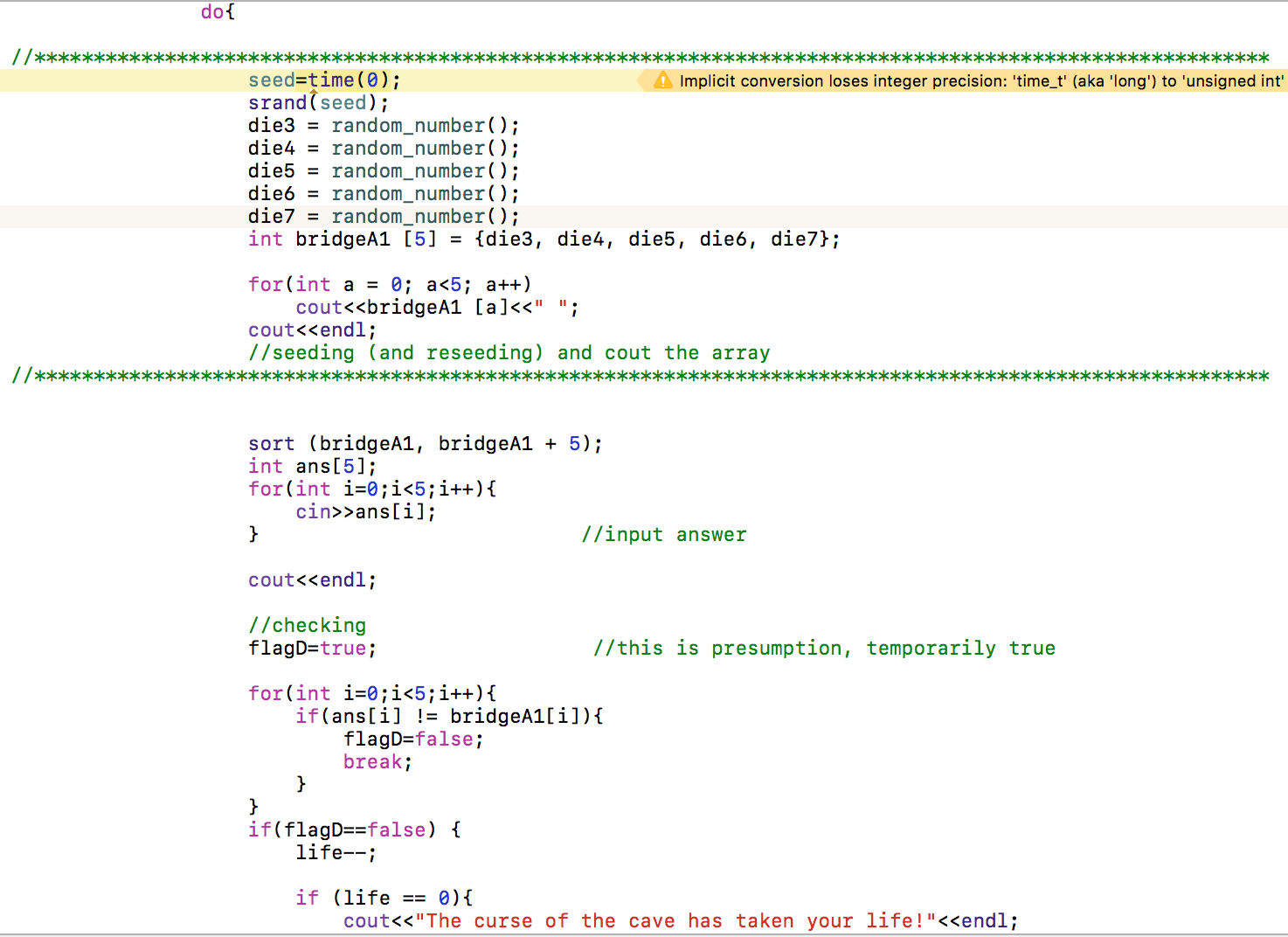
Some of the more complicated chapters in C++ are, for example, structured data, with the keyword in C++ being *struct*. It introduces abstract datatypes and even how to create them. Its applications include, pointing to a structure, passing structures to functions, and returning structures from functions. Apart from structure, there is also *class*. Although quite similar at the beginning, it allows the programmer to expand the use classes to a much greater degree than using *struct.* Classes belong to a programming style known as Object-Oriented Programming (OOP) whereas *struct* belongs to Procedural Programming. OOP comes from the need of programmers to be able to code more complicated programs, with the use of a “blueprint.” This blueprint is the class, which becomes the basis of the objects and can be used to create as many objects as desired, as long as they have the same nature, without affecting the class itself. Procedural Programming is adequate for simple programs but more advanced programs, which might use several objects, it may disrupt an already functioning code and force the user to create the “blueprints” repeatedly.

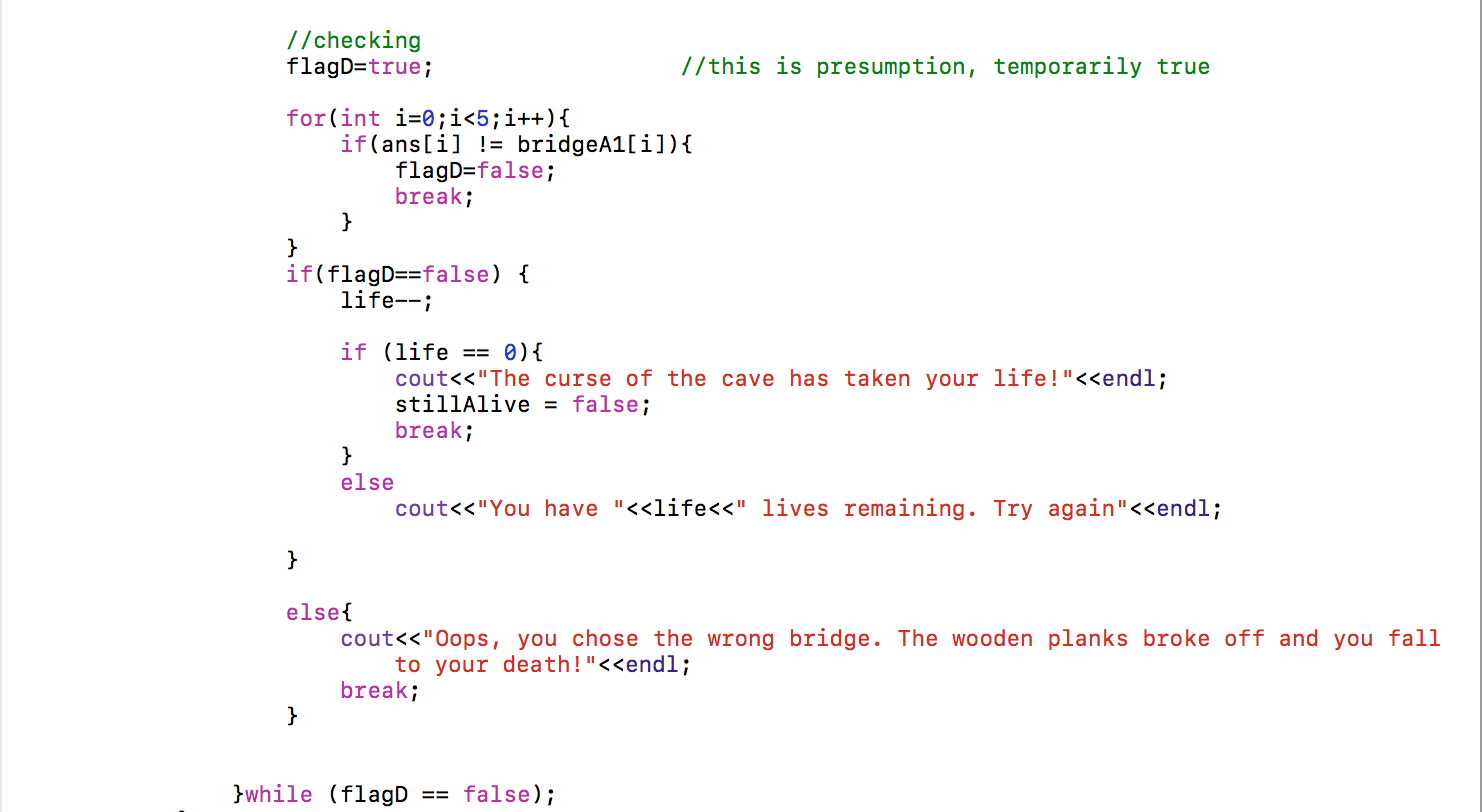
**Learning:**

**Problems overcome**

During the making of this project, some of the most difficult problems successfully overcome are making randomized numbers, checking if the player is still alive, sorting an array in descending order, creating the *sumAdder* class, and to pass the life variable and the *stillAlive* Boolean into the *RPS* (Rock – Paper – Scissors) function. All of the problems stated above will be explained in more detail below.

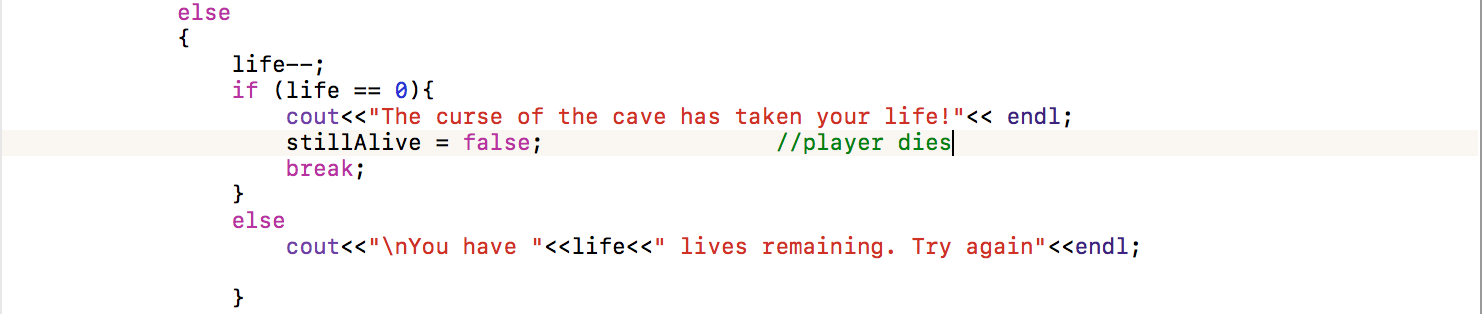
To make a number randomized, and to randomize it again if the user makes a mistake, necessitates that the seeding of the random number be inside the loop, instead of outside.

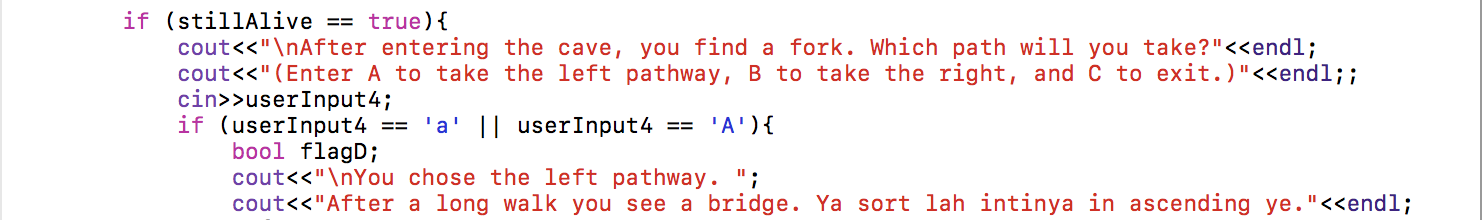




As can be seen inside the line of the asterisks, seeding of the number is included in the *do-while* loop and executed first. After the seeding is done, the program will display each of the index. If the player enters the wrong answer, the presumed *flagD* Boolean will change into false, thereby unabling the user to escape from the loop unless *life ==* 0. The loop then runs again, seeding the array with different numbers this time.

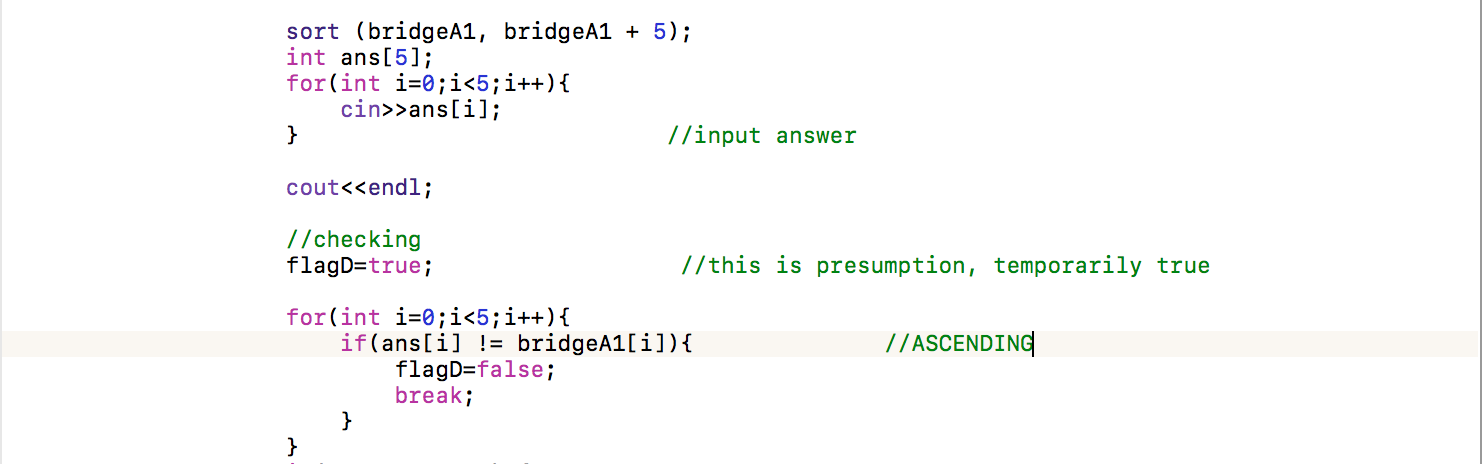
A bug occurred in which the player can still continue playing even if the life has run out. To fix this, the program needs to know that it may only continue if life is greater than 0. To check if this is true is the *stillAlive* Boolean.



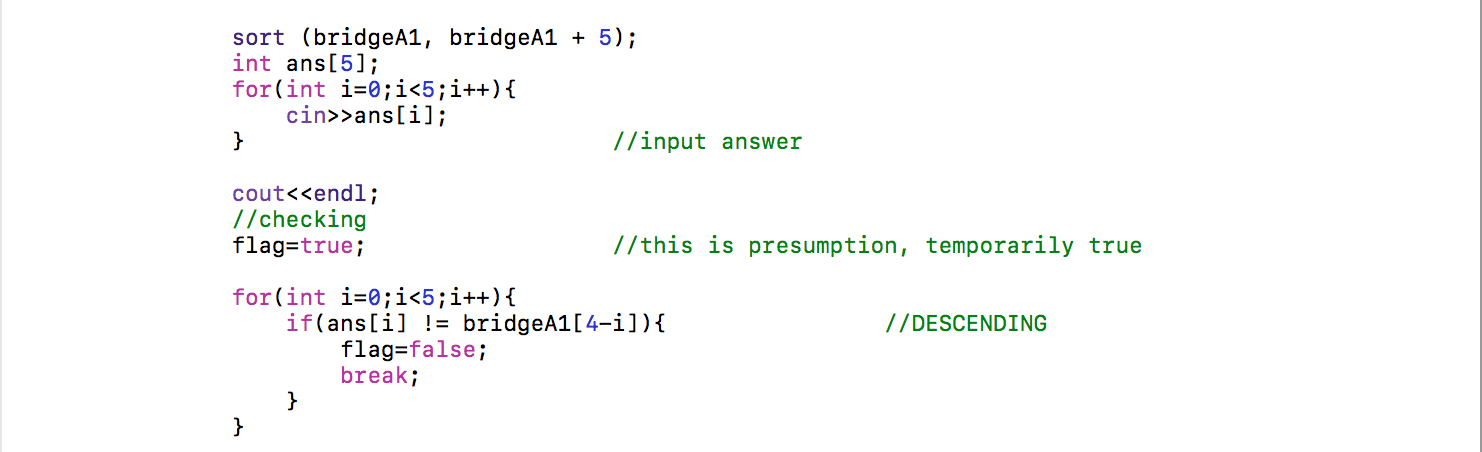


This way, the program will check first if the user still has more than one life before resuming the next section of the code, by means of only if the *stillAlive* Boolean is true. Even if the value of *life* is subtracted, *stillAlive* does not become false until it hits zero.

To sort an array, the syntax used in this program is *sort (array name), (array name) + (index amount);\** To sort the array in ascending order this is the syntax used is fairly simple.

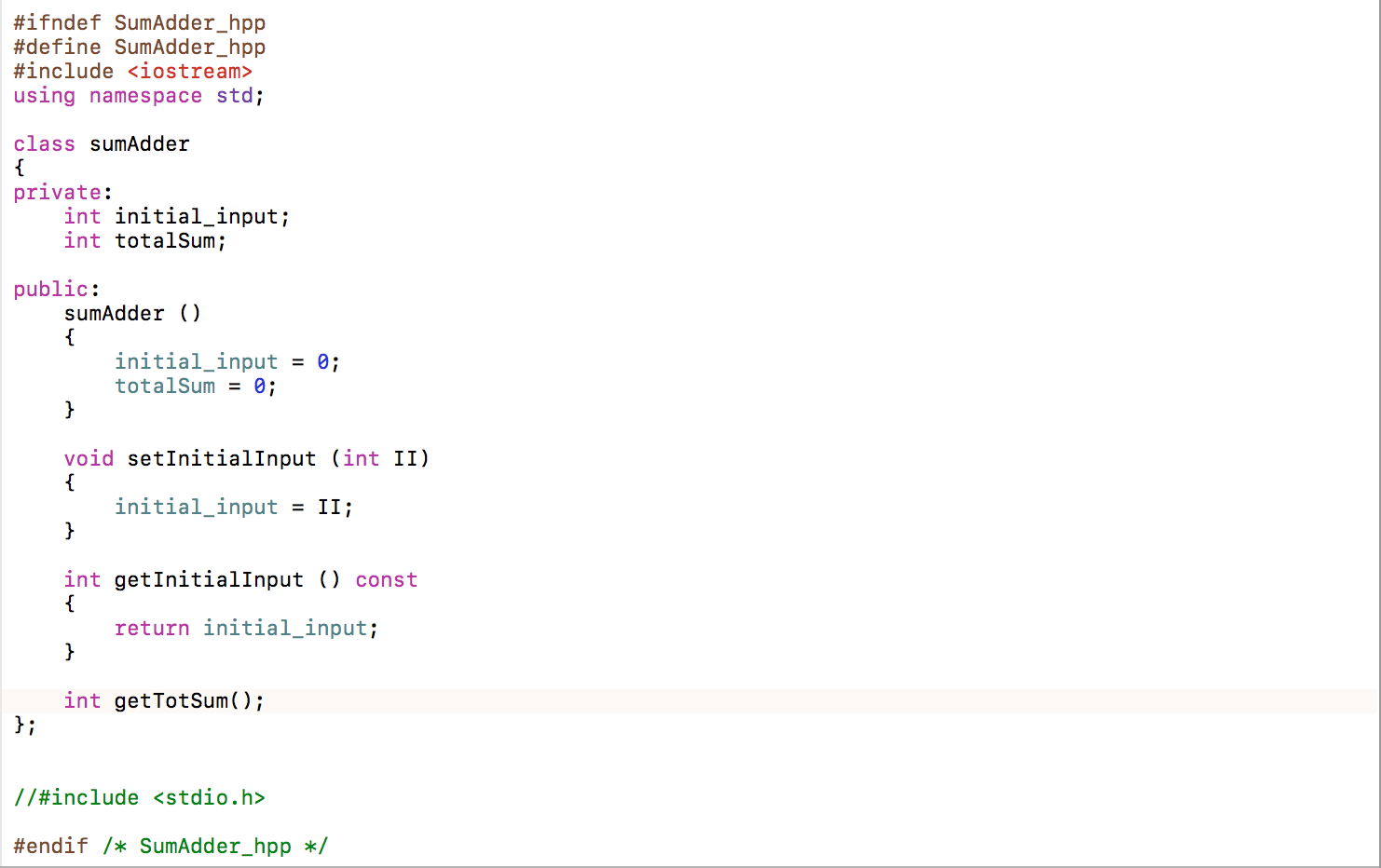


If the input at *answer* array index [i] given by the user is not equal to the *bridgeA1* array index [i], then the answer is false, thus *flagD* is false. However, since the syntax written above can only sort in ascending order, to validate if the answer inputted by the player is in descending order requires the checking to be done in reverse.

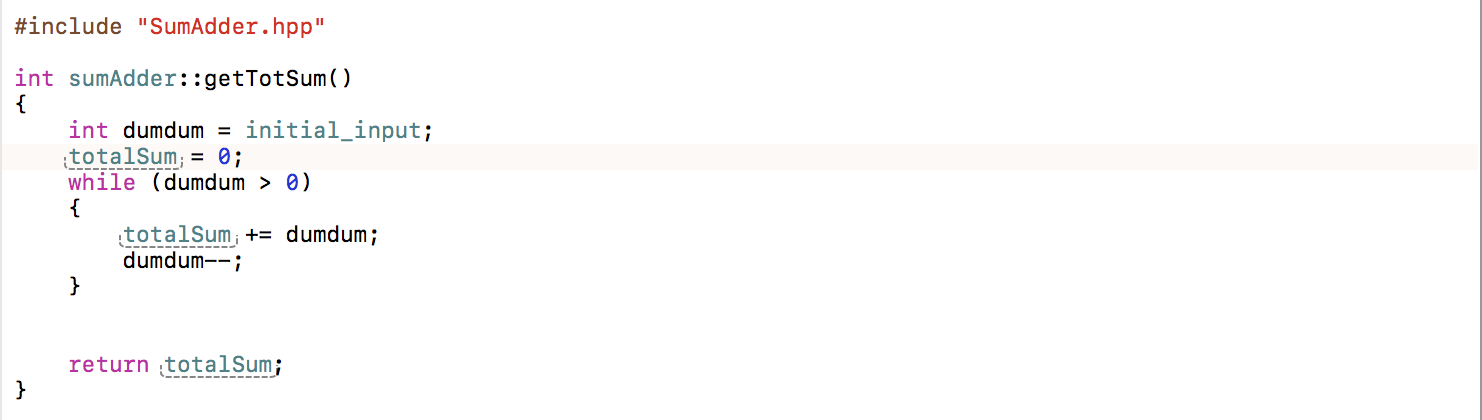


By giving the index of *bridgeA1*the value of [4-i], validating the answer will start from the biggest value number first (4 being the amount of index of an array in C++ is (n-1). The sorted array is still in ascending order though, only the checking is inverted.

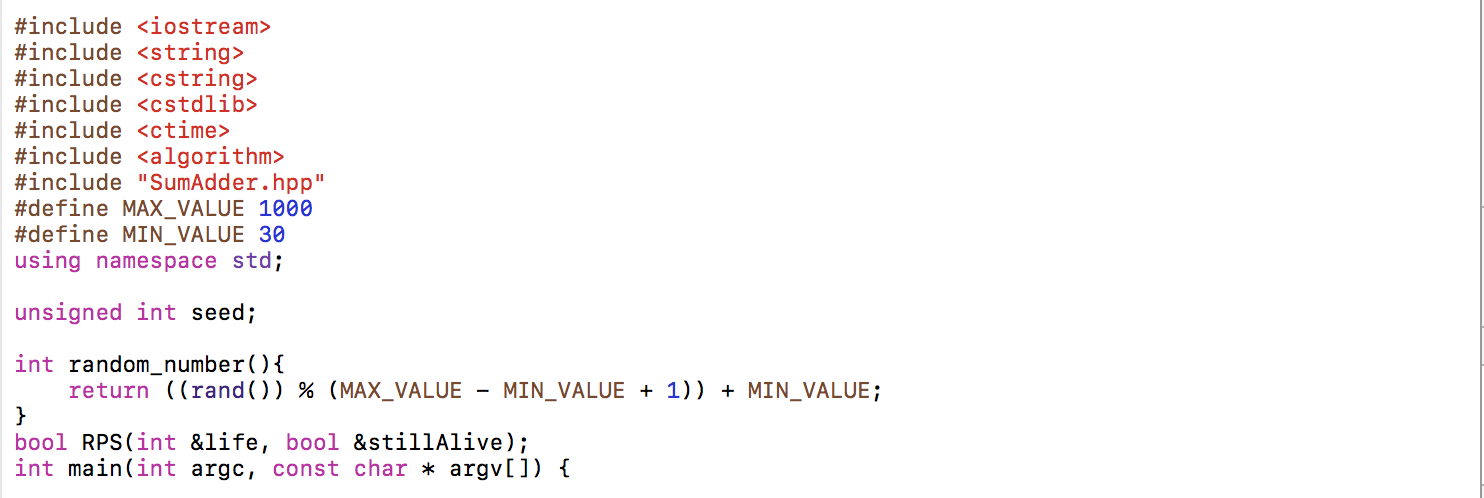
A part of the game asks the user to input an integer, and then add that number with the number before it until the number reaches zero (i.e. user inputted 4, so the next input has to be 10, derived from 4 + 3 + 2 + 1 + 0). The decision to separate this code from the main function eventually took the form of a class called sumAdder. The purpose of a class is to contain “blueprints”, and from there the objects, called instances, can be created based on that class as it acts as the basis. To make the class easy to find and create, the class is separated into a header file called SumAdder.hpp and SumAdder.hpp.

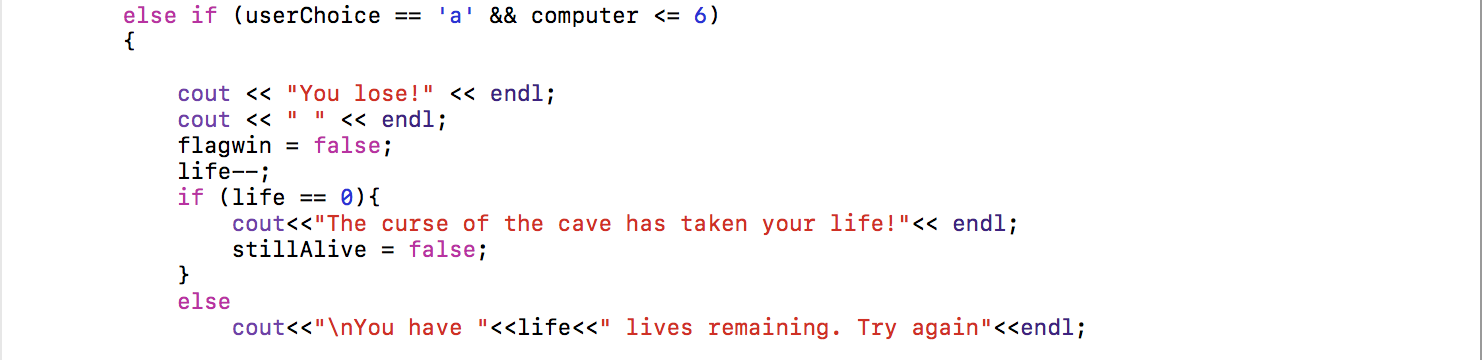


Above is the SumAdder.hpp file, which contains most of the *sumAdder* class.



This section is the SumAdder.cpp file, which contains the algorithm to add the numbers mentioned above.





The two pictures above are the function prototype for *int RPS* function and how it subtracts the player’s life with every lost round. Since the *life* variable is declared outside the *int RPS* function, it has to be able to recognize the *life* variable. To do this we need to pass that variable from the main function into the *int RPS* function by using Pass by Reference. Passing by Reference, instead of Passing by Value, is necessary because the contents of the *life* variable itself has to change. Whatever is inside of *life* is irrelevant, which is why Pass by Value is not used. What is relevant though, is that *life* remembers if *int RPS* subtracts from *life.*

**Coding**

**Main.cpp**

//

// main.cpp

// Final Project

//

// Created by Charottama Oshmar on 10/14/16.

// Copyright © 2016 Charottama Oshmar. All rights reserved.

//

#include <iostream>

#include <string>

#include <cstring>

#include <cstdlib>

#include <ctime>

#include <algorithm>

#include "SumAdder.hpp"

#define MAX\_VALUE 1000

#define MIN\_VALUE 30

using namespace std;

unsigned int seed;

int random\_number(){

return ((rand()) % (MAX\_VALUE - MIN\_VALUE + 1)) + MIN\_VALUE;

}

int RPS(int &life, bool &stillAlive);

int main(int argc, const char \* argv[]) {

char userInput1, userInput3, userInput4;

char ch; //just to enter

int userInput2, life = 3;

string status, riddle;

int n, answer;

bool stillAlive = true;

int die1; // To hold the value of rand #1

int die2; // To hold the value of rand #2

int die3, die4, die5, die6, die7;

cout<<"You are about to begin a glorious adventure! Press enter to continue"<<endl;

cin.get(ch);

cout<<"You are on a quest to find the mythical 'Daggers of Ice', which supposedly can grant its wielder a blissful and eternal life."<<endl;

cin.get(ch);

cout<< "You find a cave, with two entrances to it. Entrance A says 'Left side of the brain', while Entrance B says 'Right side of the brain."<<endl;

//cin.get(ch);

cout<<"I want to go in through Entrance... (please input your choice)"<<endl;

cin>>userInput1;

if (userInput1 == 'a' || userInput1 == 'A'){

cout<<"You have chosen Entrance A, now you must solve this equation"<<endl;

do{

seed=time(0);

srand(seed);

// Seed the random number generator.

die1 = random\_number();

die2 = random\_number();

cout << die1<< " + "<<die2<<" = ..."<<endl;

cin>>userInput2;

if (userInput2 == die1 + die2){

status = "true";

cout<<"You have succesfully made it in. "<<endl;

}

else{

do{

status = "false";

life--;

if (life == 0){

cout<<"The curse of the cave has taken your life!"<< endl;

stillAlive = false;

break;

}

else

cout<<"\nYou have "<<life<<" lives remaining. Try again"<<endl;

die1 = random\_number();

die2 = random\_number();

cout << die1<< " + "<<die2<<" = ..."<<endl;

cin>>userInput2;

if(userInput2 == die1 + die2) break;

}while(life > 0);

}

status = "true";

}

while (life > 0 && status == "false");

if (stillAlive == true){

cout<<"\nAs you venture deeper into the cave, you come across a raging underground river."<<endl;

cin.ignore();

cin.get(ch);

cout<<"The opposite end of the river is frozen, however, and you can tell you're getting close to the Daggers of Ice."<<endl;

cin.get(ch);

cout<<"There are three ancient-looking wooden bridges crossing the river. Which one would you like to take?"<<endl;

cout<<"I would like to risk my life crossing bridge... \n(Enter A, B, or C)"<<endl;

cin>>userInput3;

if(userInput3 == 'a' || userInput3 == 'A'){

cout<<"\nYou have chosen the first bridge. Along the way, you see that some planks are out of place. To reassemble them, their are numbers on them, and you can sort them in descending order."<<endl;

cin.ignore();

cin.get(ch);

bool flag;

do{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

seed=time(0);

srand(seed);

die3 = random\_number();

die4 = random\_number();

die5 = random\_number();

die6 = random\_number();

die7 = random\_number();

int bridgeA1 [5] = {die3, die4, die5, die6, die7};

for(int a = 0; a<5; a++)

cout<<bridgeA1 [a]<<" ";

cout<<endl;

//seeding (and reseeding) and cout the array

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

sort (bridgeA1, bridgeA1 + 5);

int ans[5];

for(int i=0;i<5;i++){

cin>>ans[i];

} //input answer

cout<<endl;

//checking

flag=true; //this is presumption, temporarily true

for(int i=0;i<5;i++){

if(ans[i] != bridgeA1[4-i]){ //DESCENDING

flag=false;

break;

}

}

if(flag==false) {

life--;

cout<<"You have "<<life<<" lives remaining. Try again"<<endl;

if (life == 0){

cout<<"The curse of the cave has taken your life!"<<endl;

stillAlive = false;

break;

}

}

else{

cout<<"Oops, you chose the wrong bridge. The wooden planks broke off and you fall to your death!"<<endl;

break;

}

}while (flag == false);

}

else if (userInput3 == 'b' || userInput3 == 'B')

{

bool flagB;

sumAdder sumz;

cout<<"\nBridge B is protected by very sharp stalactites. There are some scratcing however, saying you to shout a number."<<endl;

cin>>n;

sumz.setInitialInput(n);

cout<<"Therefore, n = "<<n<<endl;

do{

cout<<"The formula is 1 + 2 + 3 ... + n. Now you must solve it!"<<endl;

cin>>answer;

flagB = true;

if (answer == sumz.getTotSum()){

cout<<"Correct!"<<endl<<endl;

cout<<"The bridge you chose leads to an island on an underground lake. ";

cin.ignore();

cin.get(ch);

cout<<"At the center you find the Daggers of Ice! You attempt to use it by swinging it, but it turns out to be a fake. ";

cin.get(ch);

cout<<"The sword crumbles in your hand, and you are left in shock that this whole adventure was meaningless."<<endl;

}

else {

flagB = false;

life--;

if (life == 0){

cout<<"The curse of the cave has taken your life!"<< endl;

break;

}

else

cout<<"\nYou have "<<life<<" lives remaining. Try again"<<endl;

}

}while (flagB == false);

}

else if (userInput3 == 'c' || userInput3 == 'C')

{

cout << "\nThe bridge you crossed was relatively secure. You climbed down a cliff and you find a doorway. ";

cin.ignore();

cin.get(ch);

cout<<"After looking around, you find that the chamber is empty. Refusing to walk home empty-handed, you rubbed the walls and suddenly the wall panels move around, opening a very narrow tunnel. At the end of the tunnel, you find a golem sentinel." << endl;

cin.get(ch);

cout<<"'Greetings, Brave One' its deep voice taking you aback. 'I know what you seek, but first you must play a Rock-Paper-Scissors with me, and you must win.'"<<endl<<endl;

int flagWin = 3;

while (flagWin != 1) {

flagWin = RPS(life, stillAlive);

if (flagWin == 3){

life --;

if (life == 0){

cout<<"The curse of the cave has taken your life!"<< endl;

stillAlive = false;

}

else

cout<<"\nYou have "<<life<<" lives remaining. Try again"<<endl;

}

if (flagWin == 1){

cout<<"Congratulations, you have now become the new champion of Rock-Paper-Scissors. And now, I shall give you your prize. The Daggers of Ice!!'"<<endl;

}

}

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if (userInput1 == 'b' || userInput1 == 'B'){

bool flagC = false;

cout<<"\nYou have chosen Entrance B, but to gain entrance you must solve this riddle"<<endl;

cin.ignore();

cin.get(ch);

do{

cout<<"'When you have me, you feel like sharing me. But, if you do share me, you don't have me. What am I?'"<<endl;

getline (cin, riddle);

if (riddle == "secret" || riddle == "a secret"){

cout<<"\nThe doorway opened!";

flagC = true;

}

else

{

life--;

if (life == 0){

cout<<"The curse of the cave has taken your life!"<< endl;

stillAlive = false; //player dies

break;

}

else

cout<<"\nYou have "<<life<<" lives remaining. Try again"<<endl;

}

}while (flagC == false && life > 0);

if (stillAlive == true)

{

cout<<"\nAfter entering the cave, you find a fork. Which path will you take?"<<endl;

cout<<"(Enter A to take the left pathway, B to take the right, and C to exit.)"<<endl;;

cin>>userInput4;

if (userInput4 == 'a' || userInput4 == 'A')

{

bool flagD;

cout<<"\nYou chose the left pathway. ";

cout<<"After a long walk you see a bat. You are very shocked to see that the bat can talk.";

cin.get(ch);

cout<<"It wants you to sort some numbers, or else it will drink your blood. And remember, in ascending order!"<<endl;

cin.get(ch);

do{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

seed=time(0);

srand(seed);

die3 = random\_number();

die4 = random\_number();

die5 = random\_number();

die6 = random\_number();

die7 = random\_number();

int bridgeA1 [5] = {die3, die4, die5, die6, die7};

for(int a = 0; a<5; a++)

cout<<bridgeA1 [a]<<" ";

cout<<endl;

//seeding (and reseeding) and cout the array

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

sort (bridgeA1, bridgeA1 + 5);

int ans[5];

for(int i=0;i<5;i++){

cin>>ans[i];

} //input answer

cout<<endl;

//checking

flagD=true; //this is presumption, temporarily true

for(int i=0;i<5;i++){

if(ans[i] != bridgeA1[i]){ //ASCENDING

flagD=false;

break;

}

}

if(flagD==false) {

life--;

if (life == 0){

cout<<"The curse of the cave has taken your life!"<<endl;

stillAlive = false;

break;

}

else

cout<<"You have "<<life<<" lives remaining. Try again"<<endl;

}

else{

cout<<"Being so kind to the bat that you help soving its problem, it decides to return the favor. It shows you the way to the Daggers of Ice!"<<endl;

cin.ignore();

cin.get(ch);

cout<<"Once you touched the Daggers, the cave reveals a hidden passage to the Fountain of Life, and around it, are sacks of gold!"<<endl;

break;

}

}while (flagD == false);

}

else if (userInput4 == 'b' || userInput4 == 'B')

{

bool flagZ;

sumAdder sumz;

cout<<"\n'Inscribed along the walls, you find a message that says 'enter a number, or I will cruble while you walk under me."<<endl;

cin>>n;

sumz.setInitialInput(n);

cout<<"Therefore, n = "<<n<<endl;

do{

cout<<"The formula is 1 x 2 x 3 ... x n. Now you must solve it!"<<endl;

cin>>answer;

flagZ = true;

//cout<<sumz.multiplicResult();

if (answer == sumz.multiplicResult()){

cout<<"Correct!"<<endl<<endl;

cout<<"With heart beating so fast, you decided to go deeper into the cave.";

cin.ignore();

cin.get(ch);

cout<<"After spending more than 4 hours exploring, you begin to realize that you have been walking in circles. You gave up, and left the cave"<<endl;

}

else {

flagZ = false;

life--;

if (life == 0){

cout<<"The curse of the cave has taken your life!"<< endl;

break;

}

else

cout<<"\nYou have "<<life<<" lives remaining. Try again"<<endl;

}

}while (flagZ == false);

}

else if (userInput4 == 'c' || userInput4 == 'C')

{

cout<<"\nYou chose to leave the cave. You can start to see the light of the outside world, but suddenly an old man reveals himself from behind a big boulder.";

cin.ignore();

cin.get(ch);

cout<<"\n'Will you please play a game of Rock-Paper-Scissors with me? But please let me win, I haven't played for a very long time."<<endl;;

int flagWin = 1;

while (flagWin != 3) {

flagWin = RPS(life, stillAlive);

if (flagWin == 1){

life--;

if (life == 0){

cout<<"The curse of the cave has taken your life!"<< endl;

stillAlive = false;

}

else

cout<<"\nYou have "<<life<<" lives remaining. Try again"<<endl;

}

if (flagWin == 3){

cout<<"\nAfter losing on purpose to the old man, he suddenly stabs you. You give your best to fight back, but you are bleeding too severely and you black out!"<<endl;

}

}

}

}

}

cout<<endl;

return 0;

}

int RPS(int &life, bool &stillAlive){

char userChoice;

int computer;

int flagwin = 0;

cout << "Enter A for Rock, B for Paper, or C for scissors"<<endl;

cin >> userChoice;

userChoice = tolower(userChoice);

switch (userChoice) {

case 'a':

cout<<"you chose rock"<<endl;

break;

case 'b':

cout<<"you chose paper"<<endl;

break;

case 'c':

cout<<"you chose scissors"<<endl;

break;

default:

break;

}

srand(time(NULL));

computer = rand() % 10 + 1;

//Computer "brain"

if (computer <= 3)

{

cout << "Computer chose: Rock" << endl;

}

else if (computer <= 6)

{

cout << "Computer chose: Paper" << endl;

}

else if (computer >= 7)

{

cout << "Computer chose: Scissors" << endl;

}

//Determines winner

//Rock

if (userChoice == 'a' && computer <= 3)

{

cout << "It's a tie!\n\n";

flagwin = 2;

}

else if (userChoice == 'a' && computer <= 6)

{

cout << "You lose!\n";

flagwin = 3;

}

else if (userChoice == 'a' && computer >= 7)

{

cout << "You win!\n\n";

flagwin = 1;

}

//Paper

if (userChoice == 'b' && computer <= 3)

{

cout << "You win!\n\n";

flagwin = 1;

}

else if (userChoice == 'b' && computer <= 6)

{

cout << "It's a tie!\n\n";

flagwin = 2;

}

else if (userChoice == 'b' && computer >= 7)

{

cout << "You lose!\n";

flagwin = 3;

}

//Scissors

if (userChoice == 'c' && computer <= 3)

{

cout << "You lose!\n";

flagwin = 3;

}

else if (userChoice == 'c' && computer <= 6)

{

cout << "You win!\n\n";

flagwin = 1;

}

else if (userChoice == 'c' && computer >= 7)

{

cout << "It's a tie!\n\n";

flagwin = 2;

}

return flagwin;

}

**SumAdder.hpp**

//

// SumAdder.hpp

// Final Project

//

// Created by Charottama Oshmar on 11/3/16.

// Copyright © 2016 Charottama Oshmar. All rights reserved.

//

#ifndef SumAdder\_hpp

#define SumAdder\_hpp

#include <iostream>

using namespace std;

class sumAdder

{

private:

int initial\_input;

int totalSum;

public:

sumAdder ()

{

initial\_input = 0;

totalSum = 0;

}

void setInitialInput (int II)

{

initial\_input = II;

}

int getInitialInput () const

{

return initial\_input;

}

int getTotSum();

int multiplicResult();

};

//#include <stdio.h>

#endif /\* SumAdder\_hpp \*/

**SumAdder.hpp**

//

// SumAdder.cpp

// Final Project

//

// Created by Charottama Oshmar on 11/3/16.

// Copyright © 2016 Charottama Oshmar. All rights reserved.

//

#include "SumAdder.hpp"

int sumAdder::getTotSum()

{

int dumdum = initial\_input;

totalSum = 0;

while (dumdum > 0)

{

totalSum += dumdum;

dumdum--;

}

return totalSum;

}

int sumAdder::multiplicResult()

{

int dumdum = initial\_input;

totalSum = 1;

while (dumdum > 0)

{

totalSum \*= dumdum;

dumdum--;

}

return totalSum;

}