

|  |
| --- |
|  |
| Game Engine Creation  Ass  Ass  2019 |
| december 2019  Student Name: Charlotte Marriner  Student Number: 19023143 |



Contents

[Section 1: Variables and Operators 5](#_Toc23236754)

[Question 1: Defining User-Defined Variables (program 4) 5](#_Toc23236755)

[Program Requirements: 5](#_Toc23236756)

[Program Code: 5](#_Toc23236757)

[Program Screenshot: 5](#_Toc23236758)

[Question 2: Area of a Circle (program 8) 5](#_Toc23236759)

[Program Requirements: 5](#_Toc23236760)

[Program Code: 5](#_Toc23236761)

[Program Screenshot: 5](#_Toc23236762)

[Section 2: Conditional statements 6](#_Toc23236763)

[Question 3: Share of Haribo (program 10) 6](#_Toc23236764)

[Program Requirements: 6](#_Toc23236765)

[Program Code: 6](#_Toc23236766)

[Program Screenshot: 6](#_Toc23236767)

[Question 4: Weather Menu (program 11) 6](#_Toc23236768)

[Program Requirements: 6](#_Toc23236769)

[Program Code: 6](#_Toc23236770)

[Program Screenshot: 6](#_Toc23236771)

[Section 3: Loops 7](#_Toc23236772)

[Question 5: 10 Numbers (program 12) 7](#_Toc23236773)

[Program Requirements: 7](#_Toc23236774)

[Program Code: 7](#_Toc23236775)

[Program Screenshot: 7](#_Toc23236776)

[Question 6: Starry Output (program 13) 7](#_Toc23236777)

[Program Requirements: 7](#_Toc23236778)

[Program Code: 8](#_Toc23236779)

[Program Screenshot: 8](#_Toc23236780)

[Section 4: Functions 9](#_Toc23236781)

[Question 7: Odds and Evens (program 17) 9](#_Toc23236782)

[Program Requirements: 9](#_Toc23236783)

[Program Code: 9](#_Toc23236784)

[Program Screenshot: 9](#_Toc23236785)

[Section 5: Arrays 10](#_Toc23236786)

[Question 8: Pairs (program 22) 10](#_Toc23236787)

[Program Requirements: 10](#_Toc23236788)

[Program Code: 10](#_Toc23236789)

[Program Screenshot: 10](#_Toc23236790)

[Section 6: Strings 11](#_Toc23236791)

[Question 9: Initials (program 24) 11](#_Toc23236792)

[Program Requirements: 11](#_Toc23236793)

[Program Code: 11](#_Toc23236794)

[Program Screenshot: 11](#_Toc23236795)

[Question 10: String Manipulation (program 27) 11](#_Toc23236796)

[Program Requirements: 11](#_Toc23236797)

[Program Code: 12](#_Toc23236798)

[Program Screenshot: 12](#_Toc23236799)

[Section 7: Debugging 13](#_Toc23236800)

[Question 11: Unexpected Code (program 31) 13](#_Toc23236801)

[Program Requirements: 13](#_Toc23236802)

[Program Code: 13](#_Toc23236803)

[Program Screenshot: 13](#_Toc23236804)

[Section 8: Pointers and References 14](#_Toc23236805)

[Question 12: The Changing Pointer (program 36) 14](#_Toc23236806)

[Program Requirements: 14](#_Toc23236807)

[Program Code: 14](#_Toc23236808)

[Program Screenshot: 14](#_Toc23236809)

[Question 13: Using References (program 38) 15](#_Toc23236810)

[Program Requirements: 15](#_Toc23236811)

[Program Code: 15](#_Toc23236812)

[Program Screenshot: 15](#_Toc23236813)

[Section 9: File Handling 16](#_Toc23236814)

[Question 14: To Ten Scores (program 43) 16](#_Toc23236815)

[Program Requirements: 16](#_Toc23236816)

[Program Code: 16](#_Toc23236817)

[Program Screenshot: 16](#_Toc23236818)

[Section 10: OOP 17](#_Toc23236819)

[Question 15: Bank Account (program 47) 17](#_Toc23236820)

[Program Requirements: 17](#_Toc23236821)

[Program Code: 18](#_Toc23236822)

[Program Screenshot: 18](#_Toc23236823)

# Section 1: Variables and Operators

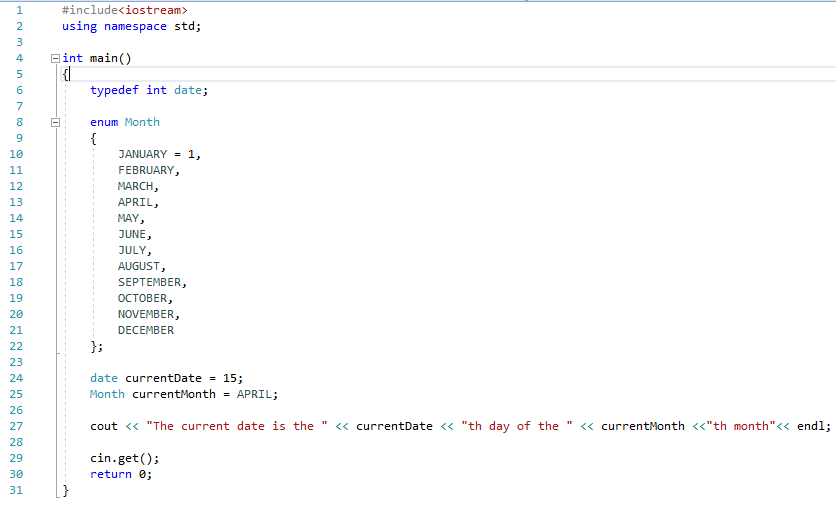
## Question 1: Defining User-Defined Variables (program 4)

### Program Requirements:

Write a program, which creates both a typedef data type named date of original data type int and an enumerated data type named Month, consisting of the 12 months with JANUARY starting at 1.

Next create a variable of data type date, with a starting value of 15, and a variable of data type Month, with its initial value to be APRIL. Output the results to the console screen.

### Program Code:



### Program Screenshot:



## Question 2: Area of a Circle (program 8)

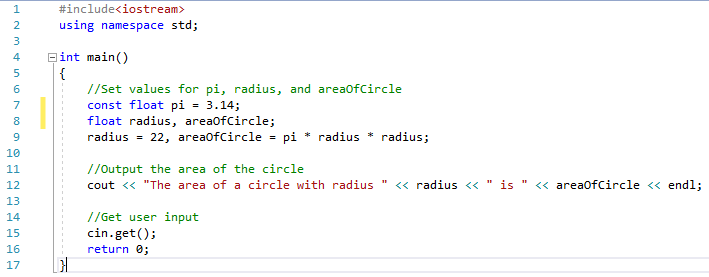
### Program Requirements:

Write a program, which creates a constant variable of float type named pi. This will hold the value 3.14. Next create two variables of floating point data type named radius and areaOfCircle. Set the starting value of radius to equal 22.0f. Calculate areaOfCircle using the following formula: Area of a circle = pi \* radius \* radius.

Output the result in the following format:

cout << “The area of a circle with radius “ << radius << “ is ” << areaOfCircle << endl;

### Program Code:



### Program Screenshot:



# Section 2: Conditional statements

## Question 3: Share of Haribo (program 10)

### Program Requirements:

Pretend that, for some reason, you are now entitled to my stash of Haribo (☹). There are 40 packets available and to make things fair, they are to be shared out equally, as in you all have to have the same number of packets.

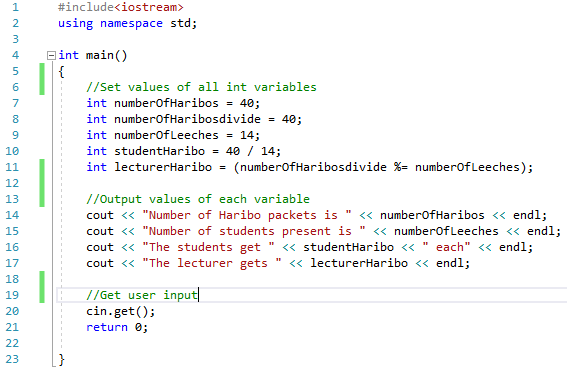
On the day when I decide to hand out the sweet sweet Haribo, only 14 students turned up (pretend that a popular video game had been released that day and many students were suddenly too “ill” to turn up).

Write a program that

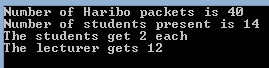
1. Calculates how many of the 40 packets of Haribo do each of the students get?
2. Calculates how many I get, which will be the remainder left after sharing the 40 packets as equally as possible

Your final answers should be 2 packets for each student, leaving 12 packets left for me. What we want to see is you coding the solution and getting the computer to do the right calculations that come up with these 2 answers. We’re not bothered about the answers themselves, similar to how many mathematics questions are more concerned with “method marks” than the final answer.

### Program Code:



### Program Screenshot:



## Question 4: Weather Menu (program 11)

### Program Requirements:

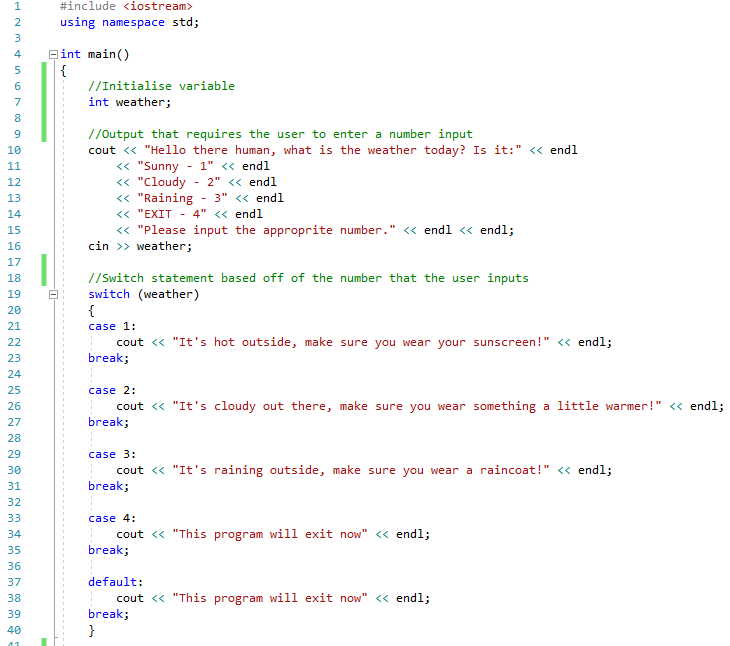
Write a program that displays a short menu such as:

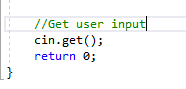
* + Please choose an option: 1. Sunny 2. Cloudy 3. Raining 4. Exit

The program should use a switch statement to display a short message appropriate to the option chosen (such as "Don't forget your sunscreen").

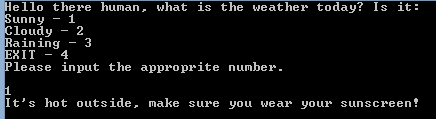
For an example of reading input from the player, take a look at Game 1: Funny Headlines in the Appendices.

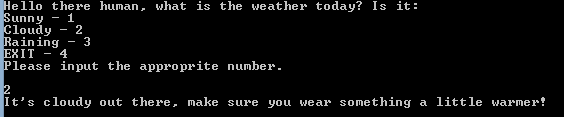
### Program Code:

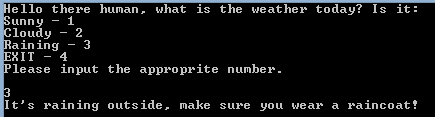


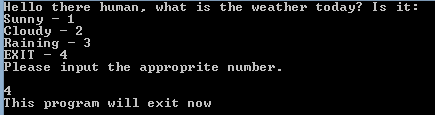


### Program Screenshot:









# Section 3: Loops

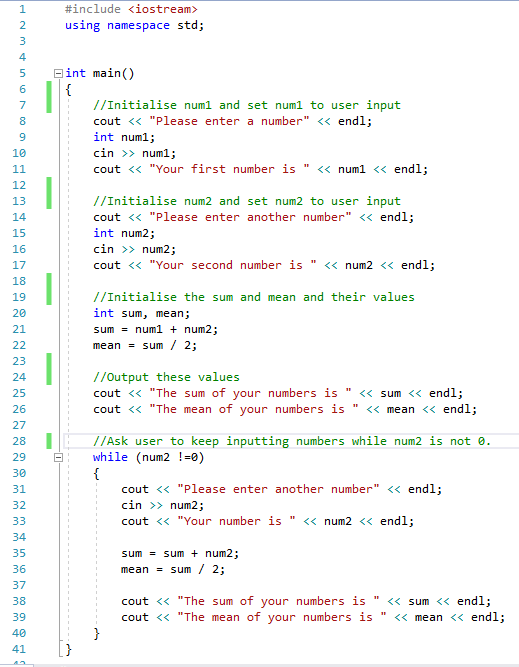
## Question 5: 10 Numbers (program 12)

### Program Requirements:

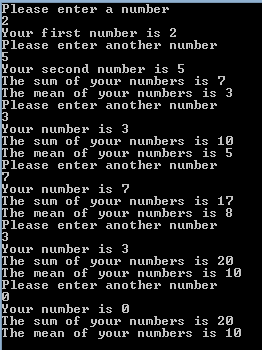
Write a program

1. That asks to user to enter 2 numbers.
   1. Using those 2 numbers, calculate the *sum* (total) of the numbers and calculate the *mean* value of those numbers.
   2. Finally, print out the *sum* and *mean* values to the console.
2. Ask the user to enter another number.
   1. Calculate the *sum* and *mean* with the current total and output in the same way as in step 1.
3. Continue to ask the user indefinitely until the user enters a value of zero.

### Program Code:



### Program Screenshot:



## Question 6: Starry Output (program 13)

### Program Requirements:

Write a C++ program

1. Which asks the user for a number *n* between 1 and 10. The program should then print out *n* lines. Each should consist of a number of stars of the same number as the current line number. For example:

Please enter a number: 5

\*

\* \*

\* \* \*

\* \* \* \*

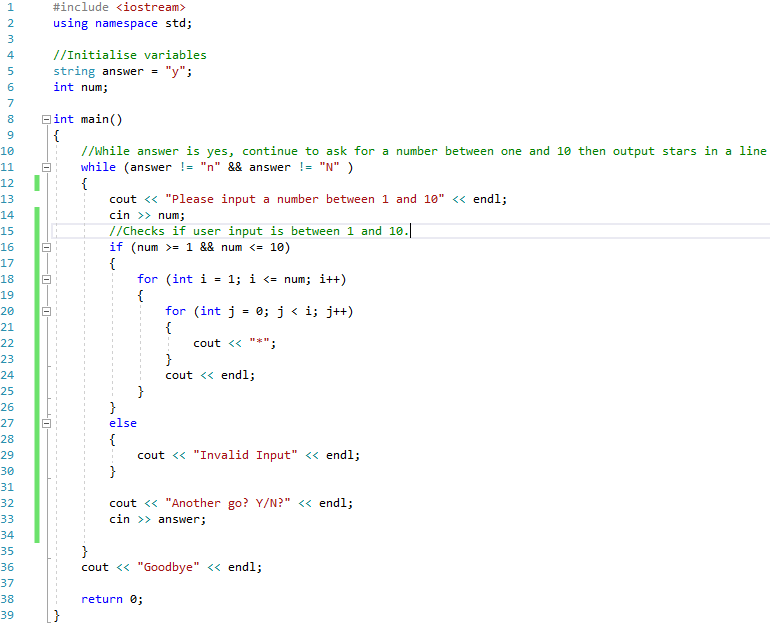
\* \* \* \* \*

1. The user should then be asked if they wish to have another go [Y or N].

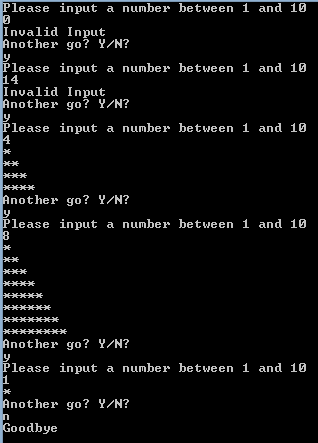
* An input of ‘Y’ will
  + Loop the player back to number 1 above,
* An input of ‘N’ will
  + Output a line of text saying “Goodbye.”
  + Pause for a brief second so the user can read the text
  + Then quit.

Hint: you will need to use nested loops!

### Program Code:



### Program Screenshot:



# Section 4: Functions

## Question 7: Odds and Evens (program 17)

### Program Requirements:

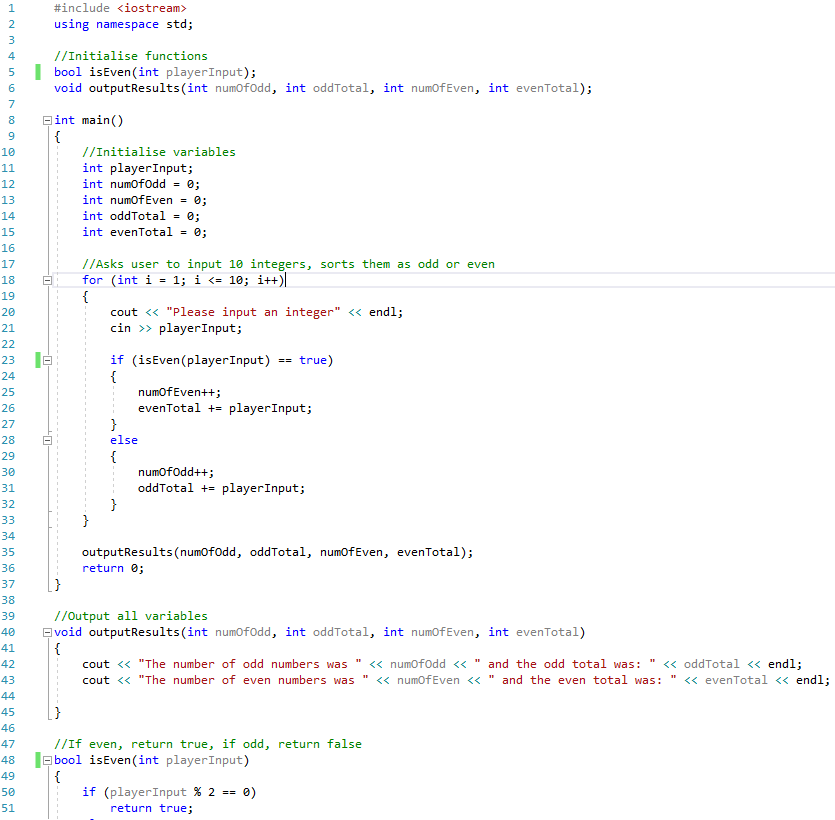
Write a program, which asks the user to input 10 integers. These values should be then passed individually to a function, which will determine whether the value is odd or even. The function should return a boolean value.

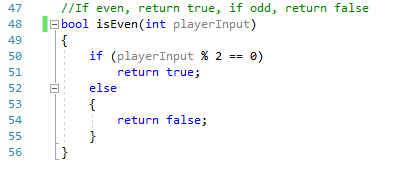
In the event of the value being odd, it should be added to an odd count, and in the event of it being even it should be added to the even count.

Your program should then call a function which will output how many odd numbers there were and the total, and in a similar manner for the even numbers. Use the following prototype for the output function:

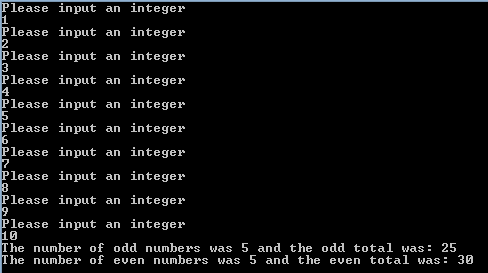
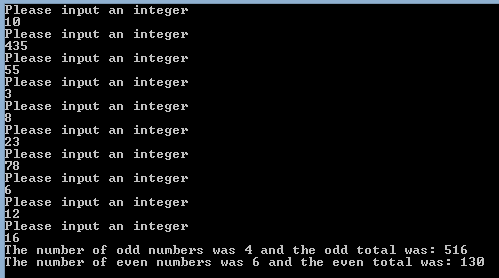
void outputResults ( int numOfOdd, int oddTotal, int numOfEven, int evenTotal );

### Program Code:





### Program Screenshot:



# Section 5: Arrays

## Question 8: Pairs (program 22)

### Program Requirements:

The player selects 2 cards (one at a time) if they match the player gets a point and the card remain face up. The game continues until all cards have been turned.

Note: This program is to demonstrate the use of multi-dimensional arrays. Ensure that you complete this program using a 2d array.

Tips

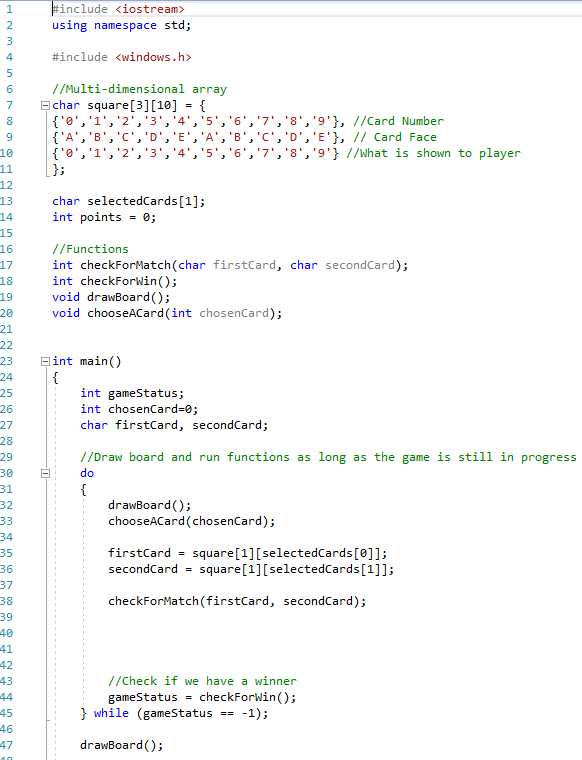
1. Take a look at how Tic-Tac-Toe (This week’s appendix game) constructed its game board. Take a similar approach for your cards.
2. Use the system(“cls”) function.
3. Use the square brackets as the cards and have a number for selection.
4. Use letters as your card faces.

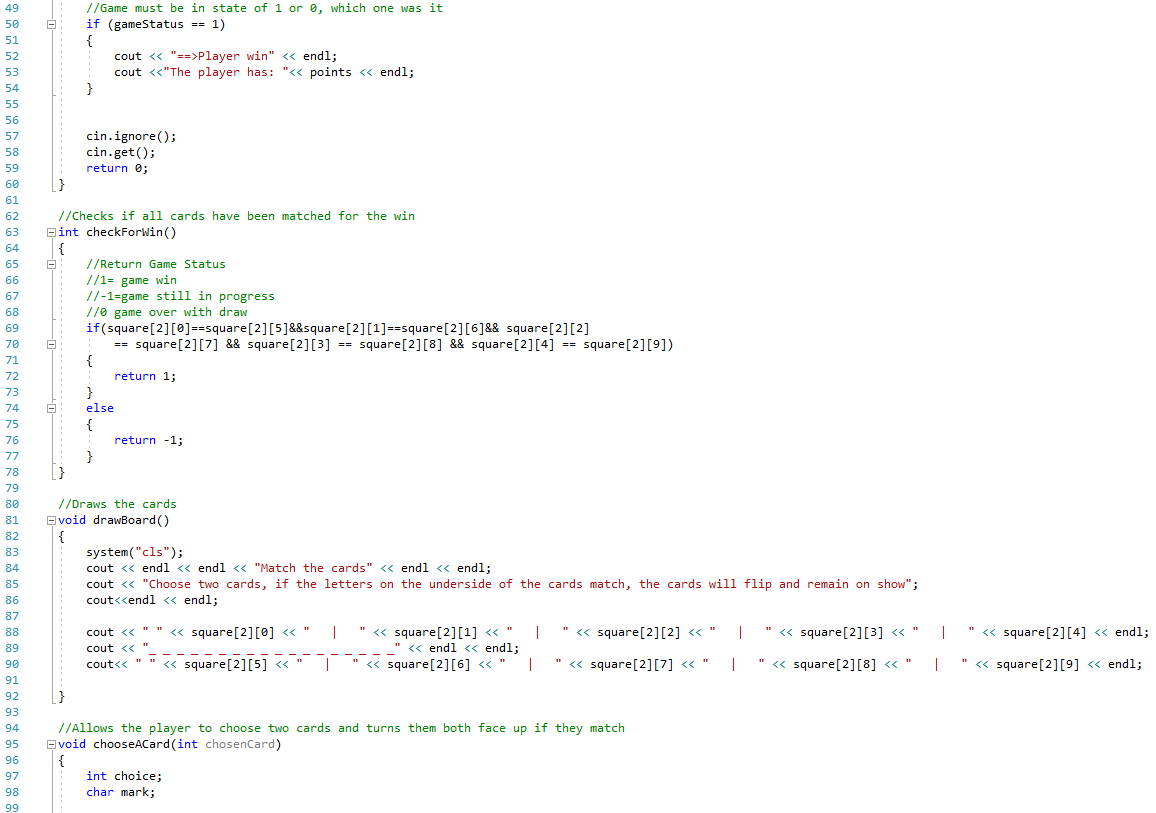
Example board: [1] [2] [3] [4] [5]

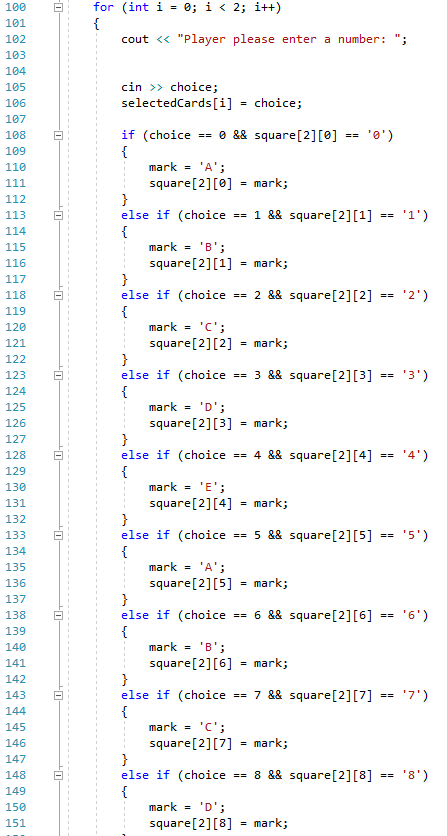
[6] [7] [8] [9] [10]

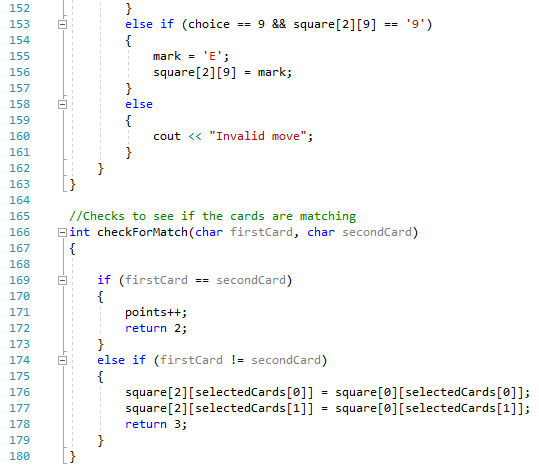
Example card faces: [A][B][C][D][E]

### Program Code:

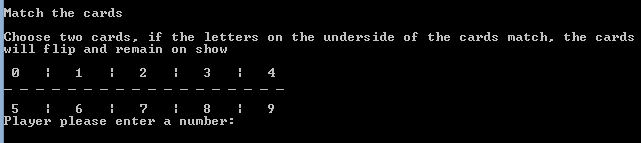


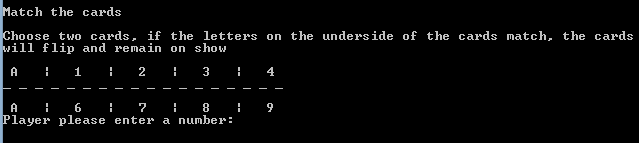


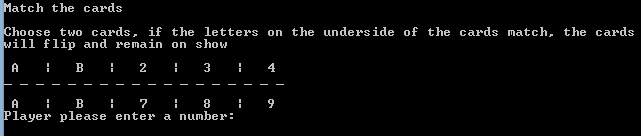


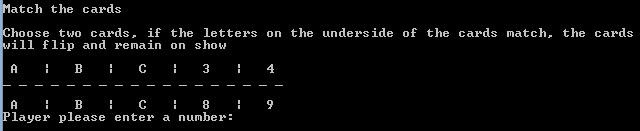


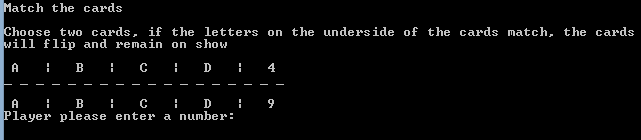
### Program Screenshot:

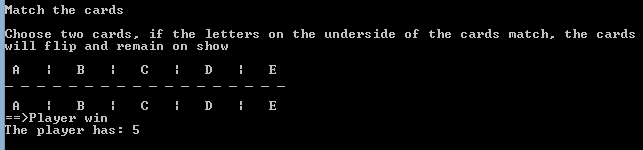












# Section 6: Strings

## Question 9: Initials (program 24)

### Program Requirements:

Write a program to prompt the user to enter in a single line their first name initial, followed by a space, their middle name initial followed by a space and the entirety of their surname. Store this in a char array.

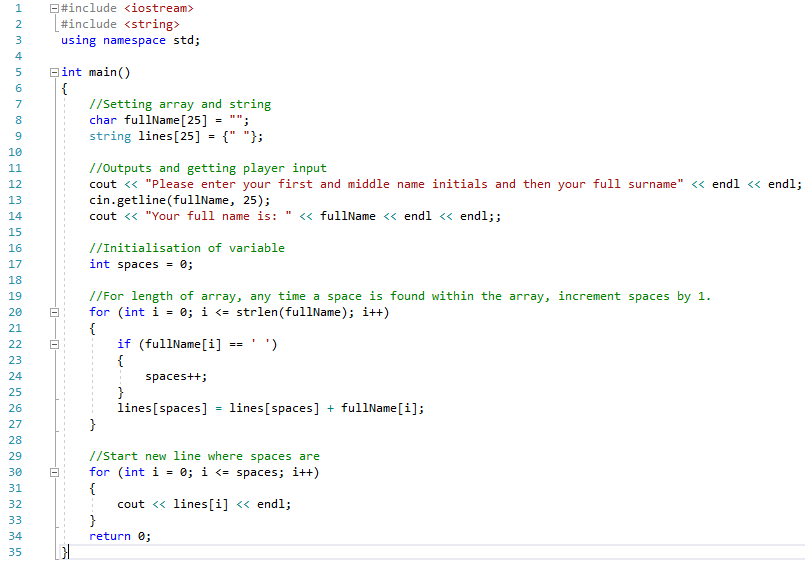
The program should then output the first initial on one line, then the middle initial on a separate line and the surname on a line of its own.

Note: This program is to demonstrate the use of char arrays. Ensure that you complete this program using a char array and that the data input by the user is stored in a single char array. You can separate out the various parts of the name into separate arrays afterwards, but the initial read from the console should put the entirety of the input into a single char array.

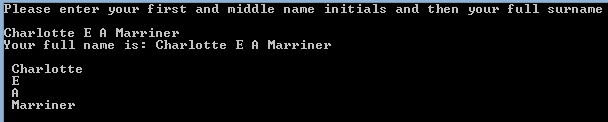
Tips

You will need to loop through your characters to find the first occurrence of a space. This should give you the information you require to be able to access the middle name initial.

### Program Code:



### Program Screenshot:



## Question 10: String Manipulation (program 27)

### Program Requirements:

Write a program to prompt the user to enter their name, which is then stored in a variable of string data type. Insert their name in between the ‘XX’ in the following string:

string testString = “Do you know who loves C++ XX does!”;

Output this string to the console. The program should then remove the two Xs and then output the string to the screen again.

The program should then ask the user to input another name (this one longer than the first). The program should then locate the first name in the test string and replace it with the new name.

Output the altered string with the second name in place of the first to the console and screenshot the entire process.

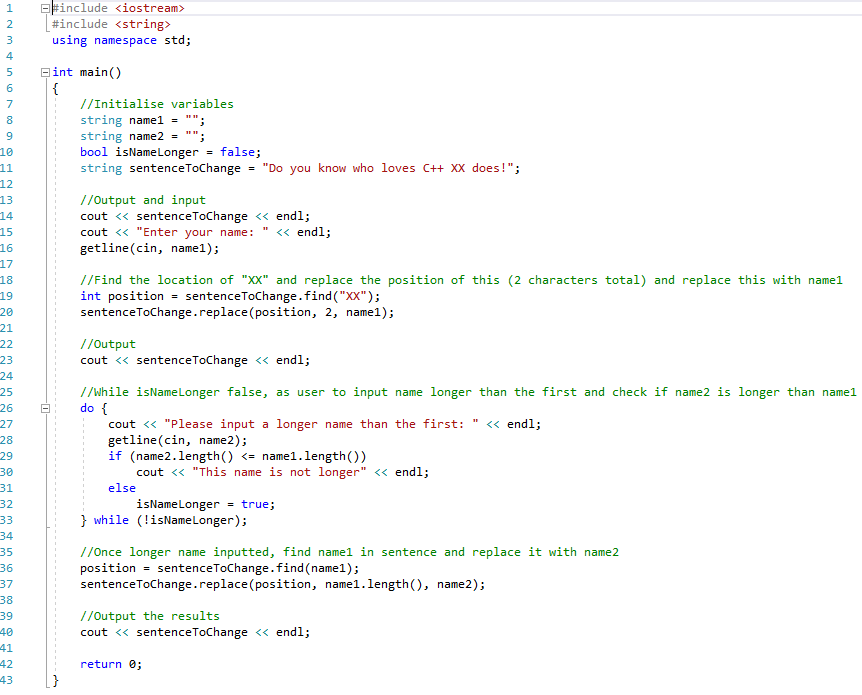
Note: The second name entered MUST be longer than the first name entered.

Note: This program is to demonstrate the use of the string data type.

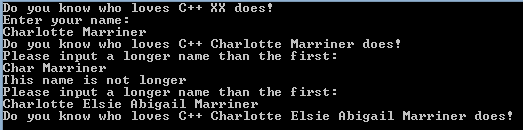
Tips

Store the first users name in its own string variable. You can then use this to locate it again when replacing it with the second name.

### Program Code:



### Program Screenshot:



# Section 7: Debugging

## Question 11: Unexpected Code (program 31)

### Program Requirements:

Enter the following program and fix all the functional errors. This program will run, but it will not give the expected results. Copy your fixed code in the relevant slot below along with a screenshot of the working program.

The expected output is:

1 is an odd number

2 is an even number

3 is an odd number

#include <iostream>

using namespace std;

int main()

{

int count = 3;

for( int i = 1; i > count; i++ )

{

cout << i;

if( count = 2 )

cout << “ is an even number” << endl;

else

cout << “ is an odd number” << endl;

}

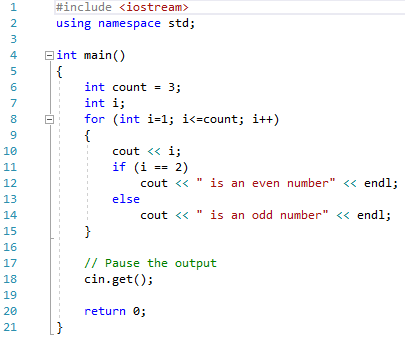
// Pause the output

cin.get();

return 0;

}

### Program Code:



### Program Screenshot:



# Section 8: Pointers and References

## Question 12: The Changing Pointer (program 36)

### Program Requirements:

Write a program that has integer variables called num1 and num2.

num1 and num2 should have values assigned by request from the user. This must be done in a function called inputDetails().This function should have the following format:

void inputDetails( int\* n1, int\* n2 );

Within the function it should ask the user to input two numbers and then populate the variables appropriately.

Within the main() function create a pointer to an int data type called pNum and point it to num1.

Write another function called outputDetails() which takes num1, num2 and pNum as parameters. This function should output the following details to the console screen:

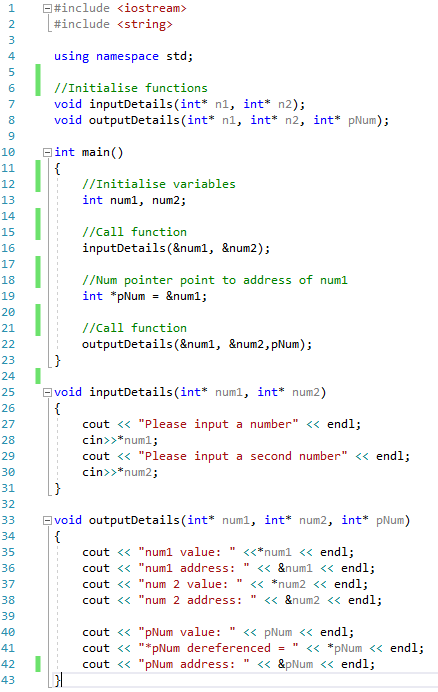
1. num1 value.
2. num1 address in memory.
3. num2 value.
4. num2 address in memory.
5. pNum value (the address it currently holds)
6. pNum dereferenced value.
7. pNum address in memory.

Ensure the output is referring to num1, num2 and pNum and not local copies. This is where the function prototype is crucial. Your output must be clear. I’d recommend outputting some text to explain each of the above.

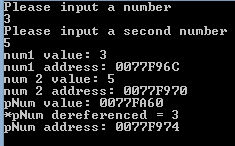
Back in the main() function reassign the pointer to point at num2 and output the same as above by calling your outputDetails() function.

Remember to set the pointer to point at nothing once you have finished with it.

### Program Code:



### Program Screenshot:



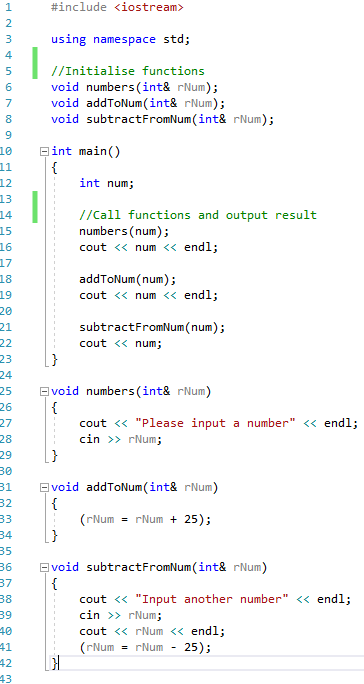
## Question 13: Using References (program 38)

### Program Requirements:

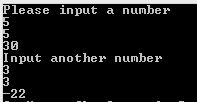
Write a program that creates an int variable called num. Next create a reference to num called rNum; All the below tasks must be done on the *reference*, with the output of num shown.

1. Ask the user to input a number and store it in num.
2. Output the value of num.
3. Add 25 to the current value stored in num.
4. Output num.
5. Ask the user to input another number and store this in num.
6. Output num.
7. Minus 25 from the current value of num.
8. Output num.

### Program Code:



### Program Screenshot:



# Section 9: File Handling

## Question 14: To Ten Scores (program 43)

### Program Requirements:

This program is to test not only your file handling ability but also how you choose to handle the data internally. Only 10 scores and names should ever be stored. A screenshot of the scores text file should also be included below. It MUST match the output expected from the screenshot of the program running.

Write a program that presents the user with the following option screen:

1. Enter a score
2. Display scores
3. Exit

If the user enters an invalid option the program should inform the user and then re-present the menu screen.

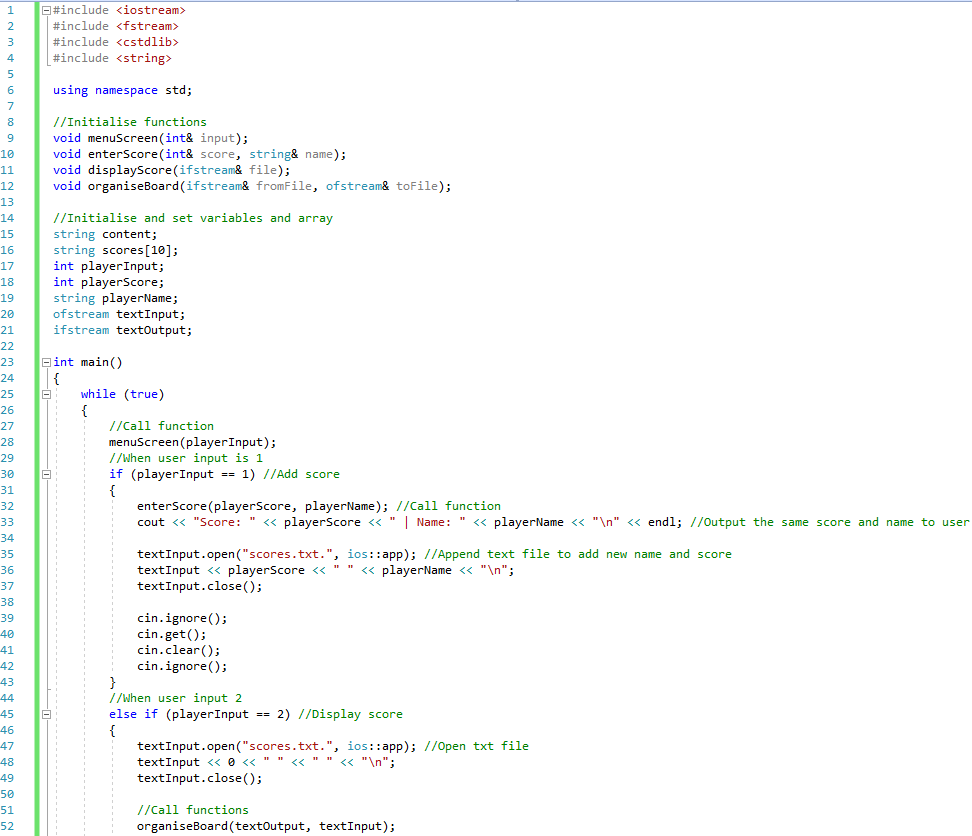
Upon entering ‘1’ the user should be asked for a score and a name. If the entered score is greater than the lowest score already in the file, or there are less than 10 scores stored this new score should be incorporated in to the top ten. This should then be saved in a text file called ‘scores.txt’. The user is then returned to the menus screen.

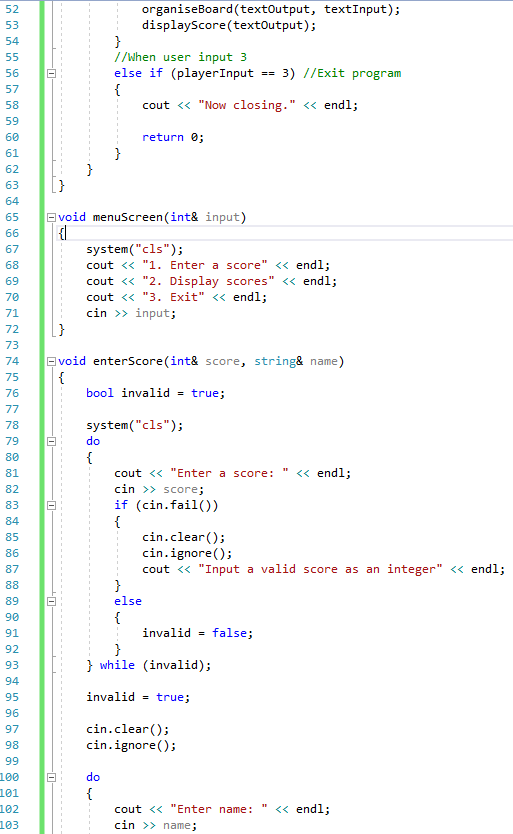
Upon entering ‘2’ the program should output a well-formatted list of scores. This should stay on screen until the user presses the return key. If there are no scores the user should be informed and then returned to the menu screen.

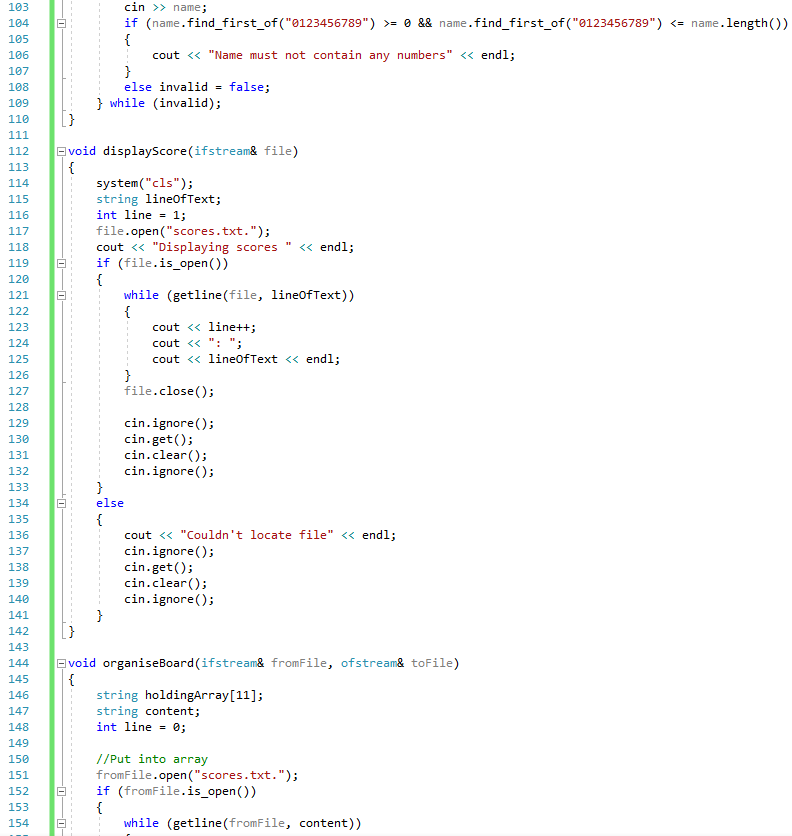
Upon entering ‘3’ the program should close. All file streams should be closed correctly.

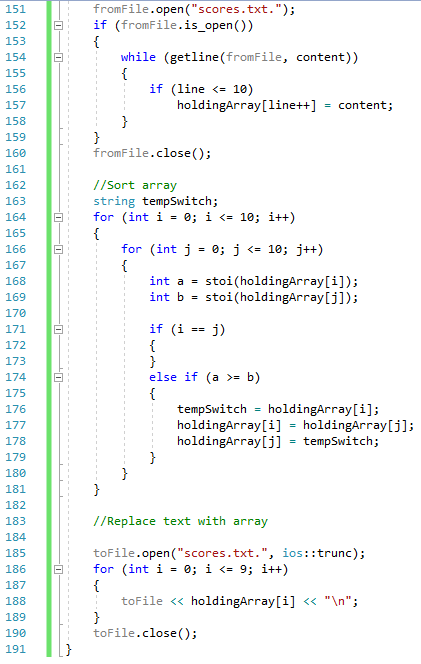
NOTE: This program must not crash. It is up to you as programmer to ensure invalid options are dealt with and that if the file doesn’t exist and the user tries to access it that they are informed, and the program returns to the menu.

### Program Code:

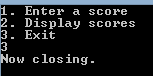
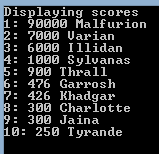
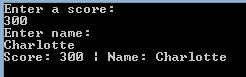








### Program Screenshot:



# Section 10: OOP

## Question 15: Bank Account (program 47)

### Program Requirements:

Write a program, which presents the user with the following menu:

1. Open a new account
2. View an account
3. Close an account
4. Exit program

Upon entering ‘1’ this second account menu is presented:

1. General account
2. Junior account
3. Savings account
4. Return to main menu

See below for the actions to be carried out when selecting from this menu.

Upon entering a ‘2’ the user is asked to enter an account number and if the account exists the following details are presented: Account number, Surname, Forename, Address, Balance, Interest rate and Account type. The user is then returned to the main menu.

Upon entering a ‘3’ the user is asked to enter an account number and if the account exists the account is closed. Closing an account removes it from the save file.

Upon entering a ‘4’ the program closes.

Within the account menu if the user enters a ‘4’ they should be returned to the main menu. On any other selection they are taken through a process of entering their details. These are to include: Surname, Forename & Address. They should be asked how much money they wish to place into the account to open it – this must be greater than 0 otherwise the process is cancelled. A unique account number is provided for the user. When the account has been opened the user is returned to the main menu.

All details must be stored on file so that they can be accessed between executions of the program. Ensure that adding a new account does not corrupt the already saved data. Likewise, when closing an account.

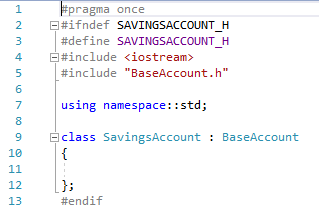
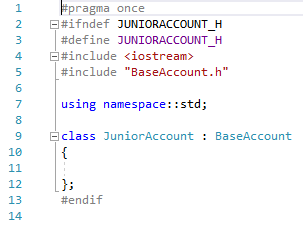
To complete this task, you should use the object-oriented approach described in this chapter. There should be a base account class with a junior account and savings account which both inherit from it. The differences in accounts will be the interest rates and account type.

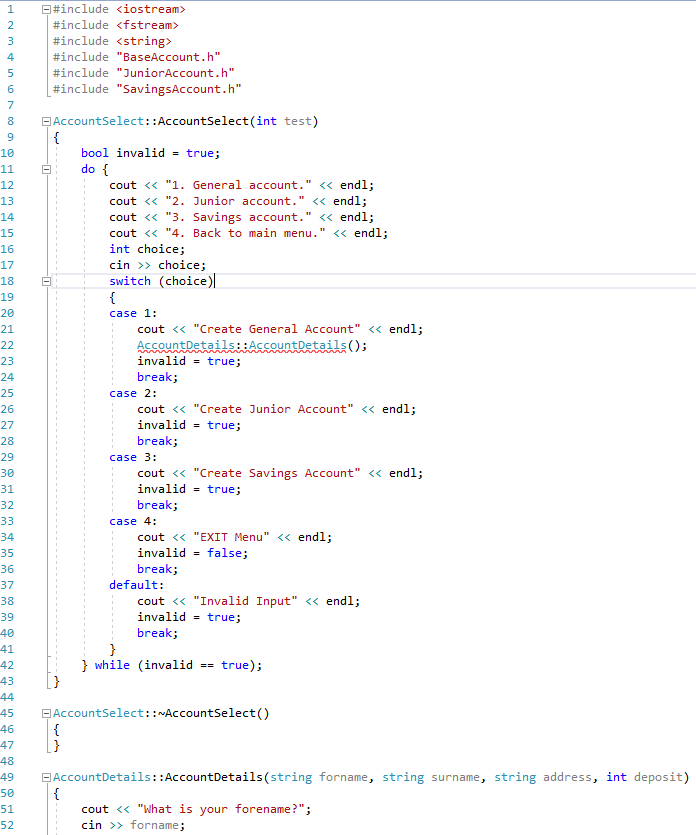
Account type should be an enum consisting of GENERAL, JUNIOR and SAVINGS.

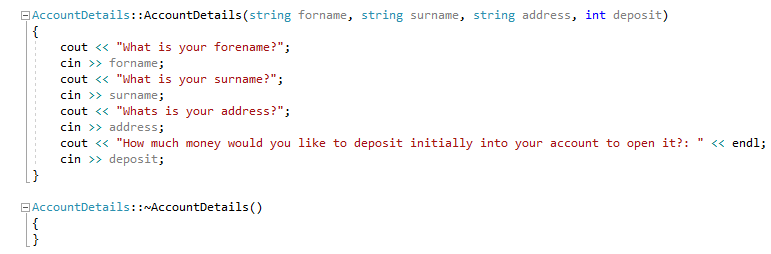
Provide screenshots of all menus and actions, along with the save file. The data shown in the file must correspond to the screenshots demonstrating it being used.

### Program Code:

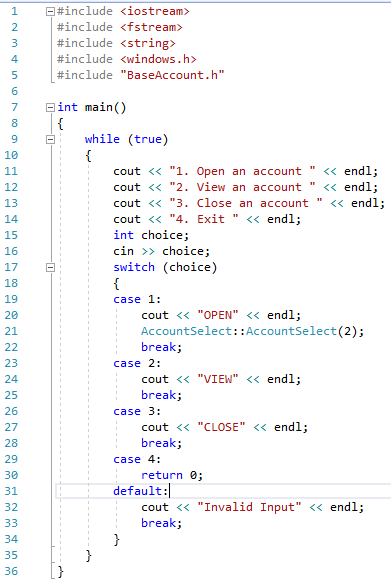














Program Screenshot:

