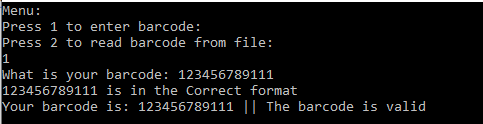
a. Your name – Joseph Rossitto

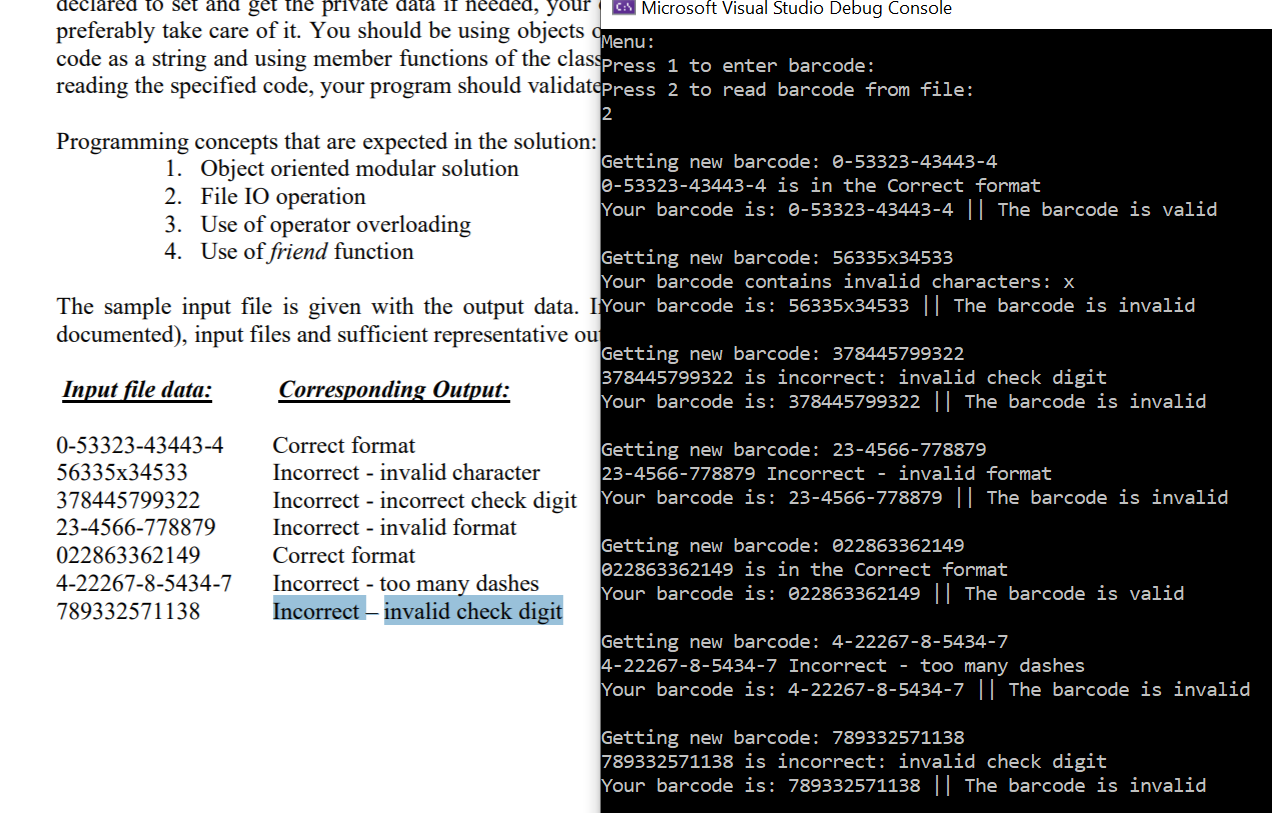
b. Id # - I don’t know how to find my Id #

c. Assignment # - Assignment 1

d. Due date – 9/17/20

e. Submitted date - 9/17/20





//a.File Name - program.cpp

//b.Author - Joseph Rossitto

//c.Date - 9/17/20

//d.Compiler Used - Visual Studio

//e.Brief Description of the file - Contains the main program

#include "BarcodeReader.h"

void main()

{

int input;

BarcodeReader barcodeReader;

cout << "Menu: " << endl;

cout << "Press 1 to enter barcode: " << endl;

cout << "Press 2 to read barcode from file: " << endl;

cin >> input;

//cout << input;

if (input == 1)

{

barcodeReader.readBarcode();

}

else

{

barcodeReader.getBarcode();

}

}

//a.File Name - BarcodeReader.h

//b.Author - Joseph Rossitto

//c.Date - 9/17/20

//d.Compiler Used - Visual Studio

//e.Brief Description of the file - Contains the documentation for barcodeReader

#pragma once

#include <iostream>

#include <string>

using namespace std;

class BarcodeReader

{

private:

string barcode = "";

string editedBarcode = "";

string validationStatement = "";

int manufacturorCode;

int productCode;

int checkDigit;

int numberSystem;

bool continueChecking;

public:

void readBarcode();

void getBarcode();

bool replaceDashes(string barcode);

bool validateBarcode();

friend ostream& operator<<(ostream& os, BarcodeReader& b);

};

//a.File Name - BarcodeReader.cpp

//b.Author - Joseph Rossitto

//c.Date - 9/17/20

//d.Compiler Used - Visual Studio

//e.Brief Description of the file - Contains the implementation for barcodeReader

#include "BarcodeReader.h"

#include <fstream>

using namespace std;

void BarcodeReader::readBarcode()

{

BarcodeReader barcodeReader;

cout << "What is your barcode: " ;

cin >> barcode;

barcodeReader.barcode = barcode;

continueChecking = true;

continueChecking = replaceDashes(barcode);

continueChecking = validateBarcode();

if (continueChecking == true)

{

barcodeReader.validationStatement = "The barcode is valid";

}

else

{

barcodeReader.validationStatement = "The barcode is invalid";

}

//cout << "The first letter is : " << barcodeFromText[0];

//Criteria 2, 3: Use of friend function and operator overloading

cout << barcodeReader;

}

void BarcodeReader::getBarcode()

{

//cout << "Your Barcode is: " << barcode << endl;

//ifstream infile("Data/Barcodes.txt");

//cout << "Getting new barcode: " << infile.get() << endl;

//infile.close();

//int sum = 0;

string barcodeFromText;

ifstream inFile;

//Criteria 1: Use file io

inFile.open("Barcodes.txt");

if (!inFile) {

cout << "Unable to open file";

exit(1); // terminate with error

}

while (inFile >> barcodeFromText) {

BarcodeReader barcodeReader;

cout << endl;

this->barcode = barcodeFromText;

barcodeReader.barcode = barcodeFromText;

cout << "Getting new barcode: " << barcodeFromText << endl;

continueChecking = true;

continueChecking = replaceDashes(barcodeFromText);

continueChecking = validateBarcode();

if (continueChecking == true)

{

barcodeReader.validationStatement = "The barcode is valid";

}

else

{

barcodeReader.validationStatement = "The barcode is invalid";

}

//cout << "The first letter is : " << barcodeFromText[0];

//Criteria 2, 3: Use of friend function and operator overloading

cout << barcodeReader;

}

inFile.close();

}

bool BarcodeReader::replaceDashes(string barcodeFromText)

{

int dashes = count(barcodeFromText.begin(), barcodeFromText.end(), '-');

//cout << dashes << endl;

if (dashes > 3)

{

cout << barcodeFromText << " Incorrect - too many dashes" << endl;

this->validationStatement = " Incorrect - too many dashes";

return false;

}

if (dashes != 3 && dashes != 0)

{

cout << barcodeFromText << " Incorrect - invalid format" << endl;

this->validationStatement = " Incorrect - invalid format";

return false;

}

this->editedBarcode = "";

for (char& c : barcodeFromText)

{

if (c != '-')

{

int barcodeCheck = (int)c;

if (!(barcodeCheck >= 48 && barcodeCheck <= 58))

{

cout << "Your barcode contains invalid characters: " << c << endl; //<< ":" << barcodeCheck << endl;

return false;

}

editedBarcode = editedBarcode + c;

}

}

//cout << "Without Dashes your barcode is: " << editedBarcode << endl;

//this->editedBarcode = barcodeFromText;

return true;

}

bool BarcodeReader::validateBarcode()

{

//cout << "Starting Validation" << continueChecking << endl;

//cout << "Without Dashes your barcode is: " << editedBarcode << endl;

int length = editedBarcode.length();

int oddSum = 0;

int evenSum = 0;

int totalSum = 0;

int remainder = 0;

if (continueChecking == true)

{

//cout << "Starting Checking" << endl;

if (editedBarcode.length() != 12)

{

cout << editedBarcode << " Incorrect - invalid format " << length << endl;

return false;

}

//1. From the right to left, start with odd position, assign the odd/even position to each digit.

int newDigit = 0;

for (int i = editedBarcode.length() - 1; i >= 0; i--)

{

if (i == 11)

{

//cout << "Edited barcode is: " << (int)(editedBarcode.at(i) - '0') << endl;

newDigit = (int)(editedBarcode.at(i) - '0');

}

//cout << "Counting " << i << endl;

else if ((i+1) % 2 == 1)

{

int currentValue = editedBarcode.at(i) - '0';

//cout << "Counting " << atoi(editedBarcode.c\_str()) << endl;

oddSum = oddSum + currentValue;

//cout << "Oddsum is: " << oddSum << endl;

}

else

{

//3. Sum all digits in even position

int currentValue = editedBarcode.at(i) - '0';

evenSum = evenSum + currentValue;

}

}

//2. Sum all digits in odd position and multiply the result by 3.

oddSum = oddSum \* 3;

//4. Sum the results of step 2 and step 3

totalSum = evenSum + oddSum;

//5. Divide the result of step 4 by 10. For example, for the above:

//cout << "OddSum is : " << oddSum << " EvenSum is : " << evenSum << " TotalSum is : " << totalSum << endl;

//totalSum = totalSum / 10;

remainder = totalSum % 10;

//cout << "Totalsum is: " << totalSum << endl;

//Subtract the remainder from 10 to get the check digit. If the remainder is 0 the

//check digit remains 0. In the example the remainder was 5; therefore, the check

// digit is 10 – 5 = 5.

int checkSum = 10 - remainder;

//checkDigit = int(editedBarcode.at(11));

//cout << "the checksum is: " << checkSum << " the check digit is: " << newDigit << endl;

if (checkSum == newDigit)

{

cout << barcode << " is in the Correct format" << endl;

return true;

}

else

{

cout << barcode << " is incorrect: invalid check digit" << endl;

return false;

}

}

else

{

//cout << "No need to check already failed" << endl;

return false;

}

}

ostream& operator<<(ostream& os, BarcodeReader& b)

{

//Criteria 2, 3: Use of friend function and operator overloading

os << "Your barcode is: " << b.barcode << " || " << b.validationStatement << endl;

return os;

}

What I learned:

I’m new to C++, my main languages are python and C# so I learned quite a lot while programming this exercise. The hardest point for me was to get the program to even start working. I had a problem in my header file that was causing an error and I was also linking my main program to my implementation and I learned that I needed to link my implementation to my header then my program to my header to work correctly.

After that I had a lot of problems with syntax and parsing the data correctly. I had to use google quite a bit on the syntax problems and the data parsing didn’t seem to work as expected. When I expected errors to be thrown I didn’t get any and when I expected int I got askii. Getting around these syntax problems took a good portion of the programming time but eventually I figured it all out.

I learned about friend functions and implementations but that was quite easy to implement, although I would have structured the program differently if I were to program it again.

Lastly I had a bit of trouble with the checksum math. I figured out that you weren’t supposed to count the 12th digit in the checksum math and I was counting the even places as odd.

After getting everything sorted I would say this was a challenging program and it took quite a bit more time then I expected it to when I read the problem description but in the end I learned a lot.