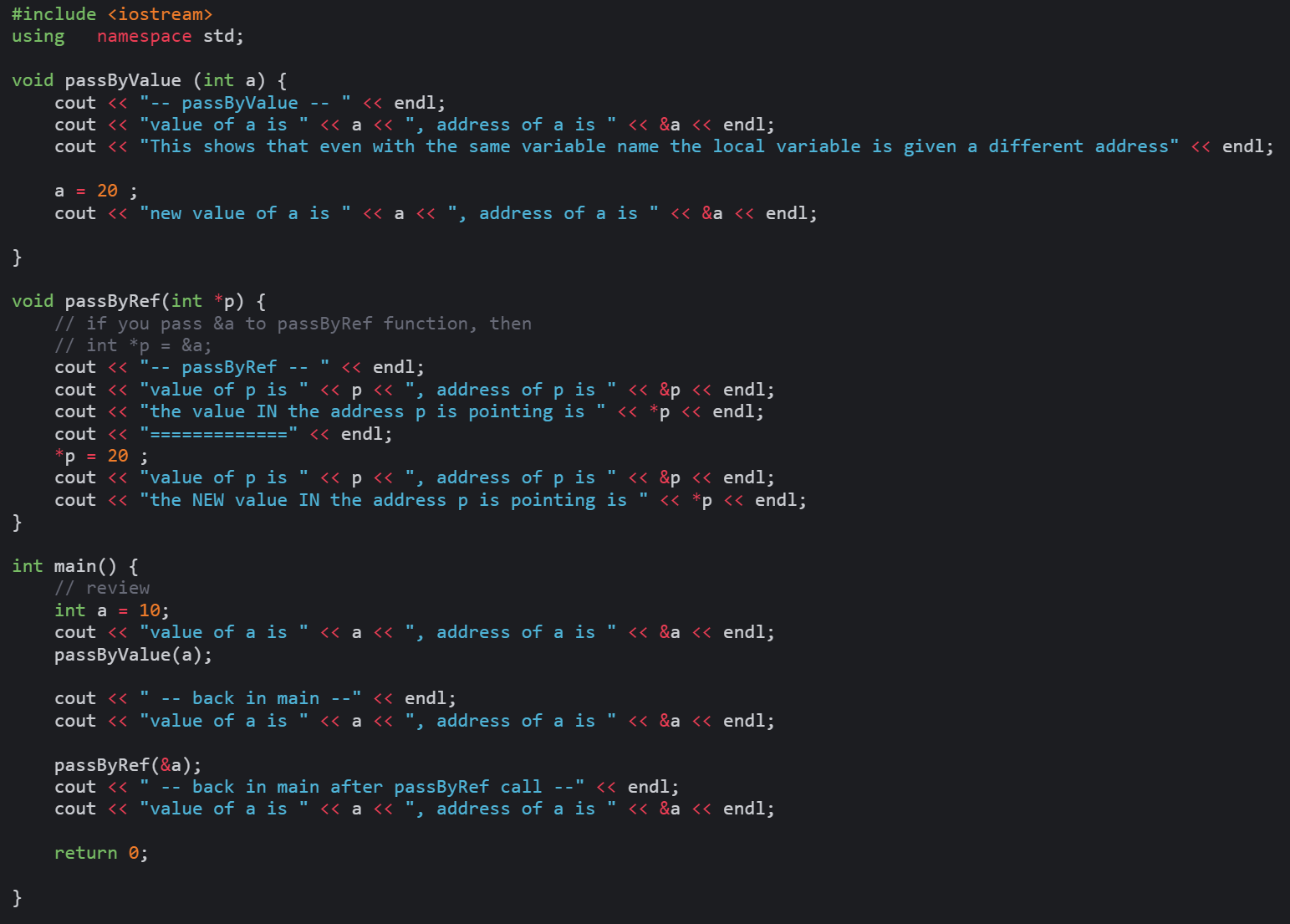
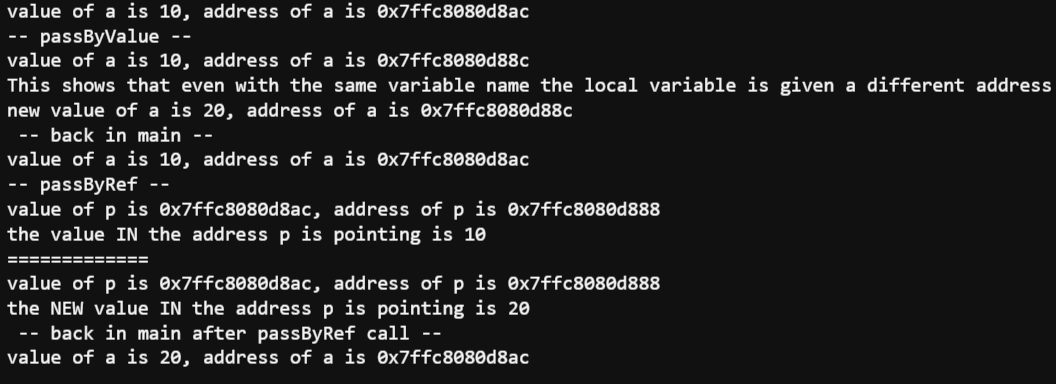
Review on basic pointers

|  |
| --- |
| #include <iostream>  using namespace std;    int main() {  // review  int a = 10;  int b = a ;  cout << "address of a is " << &a << ", address of b is " << &b << endl;  cout << "a = " << a << ", b = " << b << endl;  b = 20 ;  cout << "a = " << a << ", b = " << b << endl;    int \*p = &a; // this means p points to the address of a  cout << "address of a is " << &a << endl;  cout << "p has the value of " << p << endl;    \*p = 100 ; // \*p means you want to change the value in the address pointed by p  cout << "value of a is " << a << ", value of \*p is " << \*p << endl;  return 0;  } |





Part 1

Review on: class, function overloading, ifndef, Makefile

|  |
| --- |
| #include <iostream>    Sample output  #include <string>  #include "swapLib.h"  using namespace std;  int main() {    SwapUtil su;    // swapping two integers  int a = 10 ;  int b = 20 ;  cout << "a = " << a << ", b = " << b << endl;  su.swap(&a,&b);  Requirements:  1. Create a folder, Part1, and store all the required files inside  2. Create the swapLib.h , swapLib.cpp, and Makefile to be able to compile and execute the given code.  3. Save the given code in a file and name it TestMain.cpp  Reminder:  To recompile,  1. make clean (to remove previously created object files .o)  2. make  cout << "a = " << a << ", b = " << b << " -- after calling swap -- " << endl;      // swapping two strings  string s1 = "Johnny" ;  string s2 = "Matrix" ;  cout << "s1 = " << s1 << ", s2 = " << s2 << endl;  su.swap(&s1,&s2);  cout << "s1 = " << s1 << ", s2 = " << s2 <<" -- after calling swap -- " << endl;      // swapping two double  double d1 = 30.5 ;  double d2 = 123 ;  cout << "d1 = " << d1 << ", d2 = " << d2 << endl;  su.swap(&d1,&d2);  cout << "d1 = " << d1 << ", d2 = " << d2 <<" -- after calling swap -- " << endl;  return 0;  } |

Part 2

Review on: class, function overloading, ifndef, Makefile, array, QuickSort



Sample Output

// filename: SortLib.h

#ifndef SORTLIB\_H

#define SORTLIB\_H

class SortUtil {

public:

void quickSort(int list[], int arraySize);

int partition(int list[], int first, int last);

private:

// this function is only used by quickSort(int list[], int arraySize)

void quickSort(int list[], int first, int last);

} ;

#endif

// filename: TestMain2.cpp

#include "SortLib.h"

#include <iostream>

using namespace std;

int main()

{

SortUtil su;

const int SIZE = 9;

int list[] = {1, 7, 3, 4, 9, 3, 3, 1, 2};

su.quickSort(list, SIZE);

for (int i = 0; i < SIZE; i++)

cout << list[i] << " ";

return 0;

}

Requirements:

1. Create a folder, Part2, and store all the required files inside

2. Complete the application. Create the required SortLib.cpp and Makefile to be able to compile and execute the given code.

3. Save the given code in a file and name it TestMain2.cpp

Reminder:

To recompile,

1. make clean (to remove previously created object files .o)

2. make

Part 3

Review on: class, function overloading, ifndef, Makefile, array, QuickSort, struct

// filename: SortLib.h

#ifndef SORTLIB\_H

#define SORTLIB\_H

#include "MyStruct.h"

class SortUtil {

public:

void quickSort(int list[], int arraySize);

int partition(int list[], int first, int last);

void quickSort(MyData list[], int arraySize);

int partition(MyData list[], int first, int last);

private:

// this function is only used by quickSort(int list[], int arraySize)

void quickSort(int list[], int first, int last);

void quickSort(MyData list[], int first, int last);

} ;

#endif

// filename: MyStruct.h

#ifndef MYSTRUCT\_H

#define MYSTRUCT\_H

#include <string>

using namespace std;

struct MyData

{

int id;

string codeName;

} ;

#endif

# filename: Makefile

OBJS = TestMain3.o SortLib.o

CC = g++

SortOut: $(OBJS)

g++ -o SortOut $(OBJS)

SortLib.o:

g++ -c SortLib.cpp

TestMain2.o:

$(CC) -c TestMain3.cpp

clean:

rm -f core SortOut $(OBJS)

// filename: TestMain3.cpp

#include "SortLib.h"

#include "MyStruct.h"

#include <iostream>

#include <string>

#include <ctime>

#include <cstdlib>

using namespace std;

void display(string msg, MyData [], int);

int main()

{

SortUtil su;

const int SIZE = 9;

MyData data[SIZE];

// populating with dummy data

srand (time(NULL));

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

data[i].id = i + 1 ;

data[i].codeName = "Agent ";

data[i].codeName += 'A' + (rand()%26) ;

data[i].codeName += 'A' + (rand()%26) ;

}

display("Before", data, SIZE);

su.quickSort(data, SIZE);

display("After", data, SIZE);

return 0;

}

void display(string msg, MyData data[], int size) {

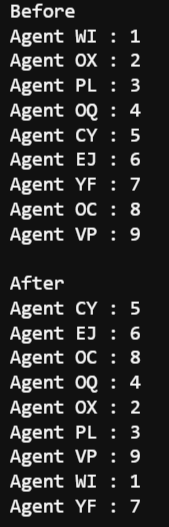
cout << msg << endl;

for (int i = 0; i < size; i++)

cout << data[i].codeName << " : " << data[i].id << endl;

cout << endl;

}



Sample output

Requirements:

1. Create a folder, Part3, and store all the required files inside

2. Complete the application. Create the required SortLib.cpp to be able to compile and execute the given code.

3. Save the given code in a file and name it TestMain3.cpp

Reminder:

To recompile,

1. make clean (to remove previously created object files .o)

2. make

Grading Criteria

|  |  |  |  |
| --- | --- | --- | --- |
|  | Part 1 | Part 2 | Part 3 |
| Fully working and all requirements followed | 1 mark | 1 mark | 2 marks |

What to submit

1. Check that all the required files are in their respective folders (e.g. Part1, Part2, Part3)
2. Remove the .o and .exe files
3. Put all the folders into 1 zip file and name it Lab4b\_yourstudentid.zip
4. Upload to D2L