

OOP

Lecture 8

02/11/2021

Quiz 2 was held in lecture 8.

Quiz 2

Task 1: Write a function to copy 2D array from source to target:

`void transform(int **src, int sR, int sC, **tar, int tR, int tC)`

Where multiple of rows and columns of source is equal to the multiple of rows and columns of target.

Solution:

```
void transform( int **src, int sR, int sC, int **tar, int tR, int tC )
{
    int i, j, m = 0, n = 0;

    for ( i = 0; i < sR; i++ )
    {
        for ( j = 0; j < sC; j++ )
        {
            tar[m][n++] = src[i][j];

            if ( n == tC )
            {
                n = 0;
                m++;
            }
        }
    }
}
```

Here, the rows and columns of source and target may or may not be the same so we have to use different variables for source and target arrays.

Task 2: Write the main function of the above program and pass two dynamic 2D arrays into transform() function.

Solution:

```
#include <cstdlib>
#include <ctime>

using namespace std;

void transform( int **src, int sR, int sC, int **tar, int tR, int tC );

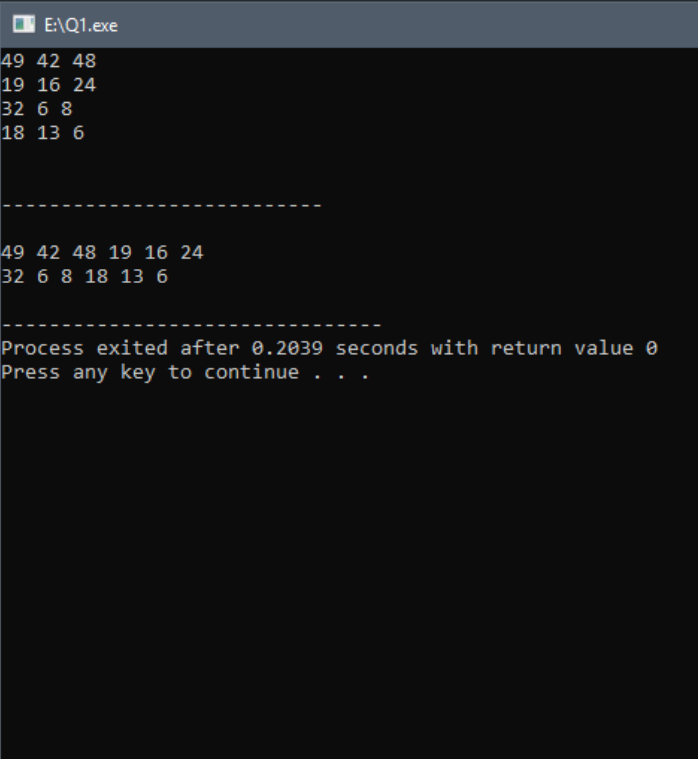
int main()
{
    int **arr1, **arr2, i, j;
    srand(time(0));

    arr1 = new int* [4];
    arr2 = new int* [2];

    for ( i = 0; i < 4; i++)
        arr1[i] = new int[3];
    for ( i = 0; i < 2; i++)
        arr2[i] = new int[6];

    for( i = 0; i < 4; i++)
    {
        for ( j = 0; j < 3; j++ )
        {
            arr1[i][j] = rand() % 50;
            cout << arr1[i][j] << ' ';
        }
        cout << '\n';
    }
    cout << "\n\n-----\n\n";
    transform( arr1, 4, 3, arr2, 2, 6 );

    for( i = 0; i < 2; i++)
    {
        for ( j = 0; j < 6; j++ )
            cout << arr2[i][j] << ' ';
        cout << '\n';
    }
}
```



The screenshot shows the output of the program. The first part displays the original 4x3 array with values ranging from 18 to 49. After the transform function call, the second part displays the resulting 2x6 array, which contains the same data flattened into two rows of six elements each. The process exits after 0.2039 seconds.

Static arrays cannot be passed to the double pointers used in the function so, we need to create Double pointers and make dynamic arrays.

Task 3: Create an array of strings (2D array) and store words with five or less than five characters in it.

```
char** convertString(char *s, int &wordCount);
```

```

#include <iostream>
using namespace std;
char** convertString(char *s, int &wordCount);

int main(){
    char sen[] = "This String is containing some words greater than five letters";
    int i, wordsCount;
    char** str;
    str = convertString(sen, wordsCount);
    for ( i = 0; i < wordsCount; i++ )
        cout << str[i] << ' ';
    for ( i = 0; i < wordsCount; i++ )
        delete[] str[i];
    delete str;
    return 0;
}

char** convertString(char *s, int &wordCount){
    int i, j = 0, k, index, count;
    char **str;
    wordCount = 0;
    for ( i = 0; s[i] != '\0'; i++ ){
        if ( s[i] == ' ' ){
            if ( count <= 5 )
                wordCount++;
            count = 0;
        }
        else
            count++;
    }
    str = new char*[wordCount];
    for ( i = 0; i < wordCount; i++ ){
        str[i] = new char[6];
        while(1){
            index = j;
            count = 0;
            for ( ; s[j] != ' '; j++ )
                count++;
            j++;
            if ( count <= 5 )
                break;
        }
        for ( k = 0; s[index] != ' '; index++, k++ )
            str[i][k] = s[index];
        str[i][k] = '\0';
    }
    return str;
}

```

E:\Q1.exe

This is some words than five

 Process exited after 0.07007 second
 Press any key to continue . . .

Task 4: Remove commas from a string (Don't use another string just manipulate the original string). There are some cases where there is a space after a comma so we need to add only one space in this condition.

Solution:

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

char* removeCommas( char* s );

int main()
{
    int i;
    char str[] = "Hello, World. This is a list, here, one,two,three,four, five";

    removeCommas(str);

    cout << str << endl;
    return 0;
}

char* removeCommas( char* s )
{
    int i, j;

    for ( i = 0, j = 0; s[i] != '\0'; i++ )
    {
        if ( s[i] == ',' )
        {
            if ( s[i+1] == ' ' )
                continue;

            s[j++] = ' ';
            i++;
        }
        s[j++] = s[i];
    }
    s[j] = '\0';

    return s;
}
```

E:\Q1.exe

Hello World. This is a list here one two three four five

Process exited after 0.1941 seconds with return value 0

Press any key to continue . . .

-- QUIZ 2 Ended --

Object Oriented Programming (OOP)

Structured programming :

Structured programming (sometimes known as modular programming) is a programming paradigm that facilitates the creation of programs with readable code and reusable functions.

Object oriented programming :

Object oriented programming allows constructing a program using a set of objects and their interactions.

Consider a situation where we want to create a program which holds the data of a whole university. What kind of things do we need to create? We need Students, Teachers, Courses etc. Then each category has its own hundreds of objects like there may be 1,000 students in a university and maybe 100 teachers. Then each entity has its own attributes which may or may not be unique like each student has a Name, Address, Section, Roll Number, etc. If we try to write this whole program in structured programming(i.e using functions), Then it is very difficult to handle that large amount of data. There may be a function which deals with the students, may be with teachers or may be with the interactions with teachers and courses etc. In short, we need to create a lot of functions and variables and arrays to handle it all. Such programs will be very difficult to write, difficult to read, difficult to extend, and very very difficult to reuse. Like we cannot reuse our program for another institute. In such cases, we transfer our way of writing programs from structured programming to Object Oriented Programming.

Object oriented programming :

In Object Oriented Programming, the program is divided into classes and objects. Object oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the C++ code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

Classes:

A class is a group having variables and functions. Variables in a class are called **Data Members**. Functions used in a program are called **Member Functions**.

In Object Oriented Programming we make objects of a class. These objects are used to access the data members and member functions of the class. A class cannot be used without its objects.

If we continue with our previous example of a university program. In OOP, we can define a class of Students, different classes of Teachers and Courses. Each class will contain its attributes as data members and all the procedures will be written in its corresponding class. We can make as many objects as we want. It will be a lot easier to handle and debug the code.

Syntax:

Classes are defined outside the main() function.

```
class class_name{  
    variables....  
    -----  
  
    Functions  
    -----  
};
```

Here, we need to consider two things: a class can not be accessed without its object, which can be defined anywhere in the program. And secondly, by default all of the data members(variables) and member functions of a class are private. It means that we cannot access them outside the class boundaries. In order to use them anywhere in the program we need to add a “public” keyword in the class. And all the things which are defined after public: can be accessed and called from anywhere in the program.

Syntax:

```
class class_name{  
    variables....  
    -----  
  
    Functions  
    -----  
    public:  
    -----  
    -----  
};
```

Here are some basic sample programs of the classes are given:

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

class Point{
    int x;
    int y;

    public:
        void setX(int x1)
        {
            x = x1;
        }
        void setY(int y1)
        {
            y = y1;
        }
        void show(void)
        {
            cout << "Point is: " << x << ',' << y << endl;
        }
};

int main()
{
    Point p1, p2;

    p1.setX(15);
    p1.setY(10);

    p2.setX(2);
    p2.setY(3);

    p1.show();
    p2.show();

    return 0;
}
```

E:\Q1.exe
Point is: 15,10
Point is: 2,3

Process exited after 0.09858 seconds with return value 0
Press any key to continue . . .

The data members i.e x, y are private and the functions are public. Here, an important thing to notice is that we didn't pass the x and y parameters to the show() function. It is because when we call the show()

function with its object i.e p1.show() then all the attributes of p1 (in this case x and y) can be accessed in the class.

Constructors:

A constructor in C++ is a special function that is automatically called when an object of a class is created. It cannot be called after the definition of the object. It can only be called once with one object.

Two important points about constructors are:

- The name of a Constructor should be same as the class name
- Constructors do not have any return type.

```

#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

class Point{
    int x;
    int y;

    public:
        Point(int x1, int y1)
        {
            setX(x1);
            setY(y1);
        }

        void setX(int x1)
        {
            x = x1;
        }
        void setY(int y1)
        {
            y = y1;
        }
        void show(void)
        {
            cout << "Point is: " << x << ',' << y << endl;
        }
};

int main()
{
    Point p1(10, 15);
    Point p2(2, 3);

    p1.show();
    p2.show();

    return 0;
}

```

E:\Q1.exe

Point is: 10,15
Point is: 2,3

Process exited after 0.06741 seconds with return code 0
Press any key to continue . . .