NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
CL 218–DATA STRUCTURES LAB
Lab Session 03

Instructors: Mr. Irfan , Ms. Mubashra

Outline

- **Safe Array**
- **Jagged Array**

Dynamic Safe Array:

In C++, there is no check to determine whether the array index is out of bounds. During program execution, an out-of-bound array index can cause serious problems. Also, recall that in C++ the array index starts at 0. Safe array solves the out-of-bound array index problem and allows the user to begin the array index starting at any integer, positive or negative.

Sample code:

```
#include<iostream>
#include<fstream>
#include <stdio.h>
#include <string.h>
#include<cstdlib>

using namespace std;

class ArrayIndexOutOfBoundException {
public:
  ArrayIndexOutOfBoundException()
        : message ("Array Index Out of Bound") {}
  ~ArrayIndexOutOfBoundException() { /*delete message; */}
  const string what() const { return message; }
private:
  char *message;
};

class DynamicSafeArray{

   private:
        int *Data;
        int size;


  public:
        DynamicSafeArray(){

                Data=NULL;
```

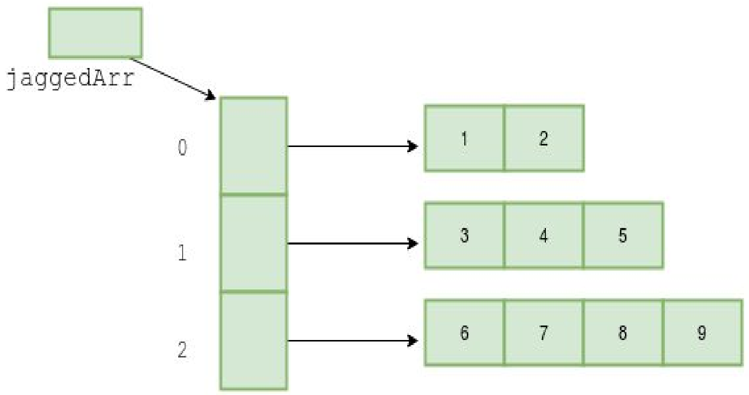
```
                        size=0;


            }

            DynamicSafeArray(int n){

                        size=n;
                        Data= new int[size];
                        memset(this->Data, 0, sizeof(int)*size); }

    DynamicSafeArray(const DynamicSafeArray & rhs){

                         this->size= rhs.size;
                         this->Data= new int[size];
                         memcpy(this->Data,rhs.Data,(sizeof(int)*rhs.getSize()));
}
      ~DynamicSafeArray()
     {
            if(Data !=0)
            {
                        cout<<"bye--"<<endl;
                        delete [] Data;
                        Data=0;
                        size=0; }
}
  void ReSize(int nSize){

                        if (size != nSize )
                        {
                         int * temp= new int[size];

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

                                    temp[i]= *(Data+i);
                            }
                            delete[] Data;
                            Data=0;

                            Data = new int[nSize];
                            memset(this->Data, 0, sizeof(int)*nSize);
                                    for(int i=0; i<size ; i++){

                                    *(Data+i)=temp[i];
                            }
                            size= nSize;
                            delete [] temp;
                            temp=0;
  }

}
unsigned int getSize() const {
    return size;
```

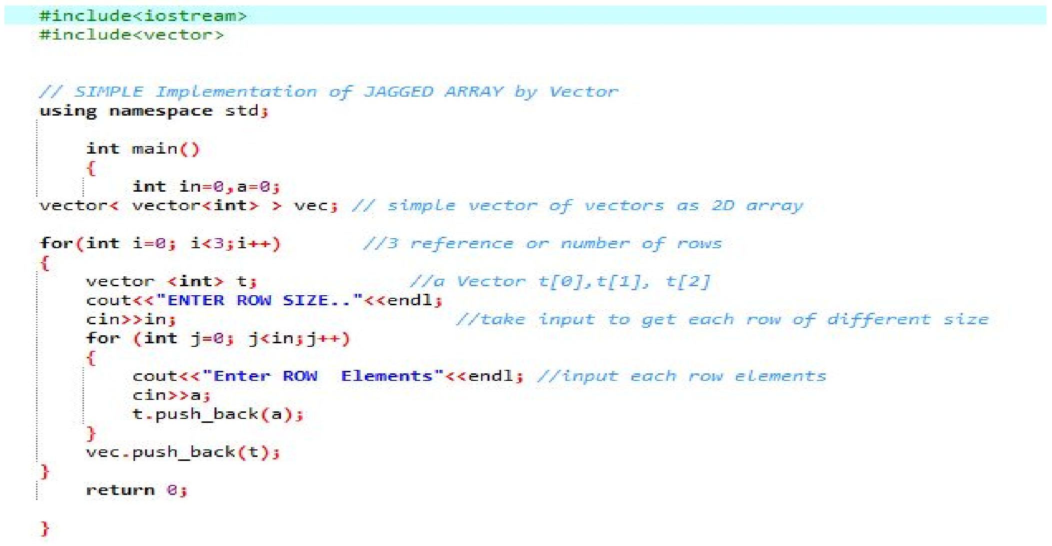
```
   }
  DynamicSafeArray& operator=( DynamicSafeArray & rhs)
     {
        if (this  != &rhs)
        {

        int s=rhs.getSize();
        this->size=s;
        this->Data= new int[s];
        memcpy(this->Data,rhs.Data, sizeof(rhs.Data));
        }
        return (*this);
     }
//lval
   int& operator[](unsigned int i){

    return *(Data+i);}

   //rval
  const int& operator[](unsigned int i) const {
  return *(Data+i);}

friend istream& operator >> (istream& infile, DynamicSafeArray & rhs)
{
  for (int count=0;count<rhs.size;count++)
              infile>>rhs.Data[count];
    return infile;
}

friend ostream& operator << (ostream& outfile,DynamicSafeArray & rhs)
{
  for (int count=0;count<rhs.size;count++)
              outfile<<rhs.Data[count];
    return outfile;
}

};

int main()
{

  DynamicSafeArray DSA1(10);

  cout << "Enter an Array DSA1: ";
  cin >> DSA1;
  cout << DSA1;
   std::ofstream out_file;
   out_file.open("output.txt");
   out_file << DSA1<< endl;
   out_file.close();
return 0;
}
```

Jagged Array:

A jagged array is also a multi-dimensional array, comprising arrays of varying sizes as its elements (rows). It also referred ragged array.

Jagged array in memory:





```
#include<iostream>
#include<vector>

// SIMPLE Implementation of JAGGED ARRAY by Vector
using namespace std;

    int main()
    {
        int in=0,a=0;
vector< vector<int> > vec; // simple vector of vectors as 2D array

for(int i=0; i<3;i++)        //3 reference or number of rows
{
    vector <int> t;              //a Vector t[0],t[1], t[2]
    cout<<"ENTER ROW SIZE.."<<endl;
    cin>>in;                         //take input to get each row of different size
    for (int j=0; j<in;j++)
    {
        cout<<"Enter ROW  Elements"<<endl; //input each row elements
        cin>>a;
        t.push_back(a);
    }
    vec.push_back(t);
}
    return 0;

}
```

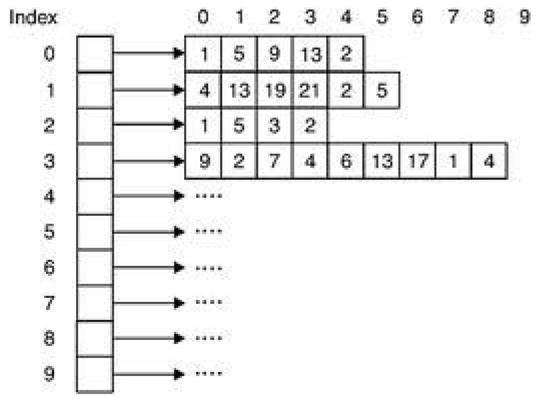
Lab Task:

Question1:

Your task is to create a class DSA in DSA.h file and declare two functions 'insertion' and 'searching' in it in addition to the functions discussed in the rule of three. Now inherit two classes 'Ordered Dynamic Safe Array' and 'Unordered Dynamic Safe Array' and implement both the functions for these classes.

Question2:

Write a program that will read n integers from the keyboard and place them in a jagged array as shown in the following diagram:


Index
0 1 2 3 4 5 6 7 8 9
0
1
2
3
4
5
6
7
8
9
1 5 9 13 2
4 13 19 21 2 5
1 5 3 2
9 2 7 4 6 13 17 1 4
....