جامعة المنارة
كلية الهندسة
قسم الهندسة المعلوماتية

مقرر الخوارزميات و بنى المعطيات1

جلسة العملي الرابعة

(الفصل الثاني 2023-2024)

Queue الرتل

تمرين 1:

اكتب برنامج يقوم بتعريف رتل مكون من الأرقام التالية:

7 77 5 55 واختبر الرتل فيما إذا كان فارغا قبل وبعد إدخال الأرقام ومن ثم قم بحذف أول رقمين ثم اطبع عناصر الرتل

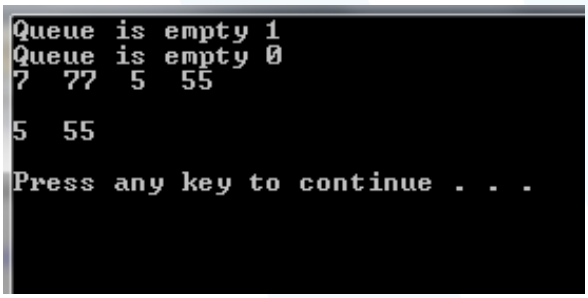
الحل

```
#include "stdafx.h"
#include<iostream>
using namespace std;
#ifndef QUEUE
#define QUEUE

const int QUEUE_CAPACITY=128;
typedef int QueueElement;

class Queue
{ /****  function members ****/
public:
Queue();
bool empty( ) const ;
void addQ(const QueueElement & value);
QueueElement front( ) const;
void removeQ();
void display() const ;

/****  data members ****/
```

```
private:
        QueueElement myArray[QUEUE_CAPACITY];
        int myFront,myBack;
}; //end of class declaration

inline Queue::Queue()
{ myFront=myBack=0; }

inline bool Queue::empty( ) const
{ return (myFront==myBack)  ;}

void Queue::addQ(const QueueElement & value)
{    int newBack=(myBack+1)%QUEUE_CAPACITY;
  if (newBack==myFront)
          cout<<"**** QUEUE IS FULL ****"<<endl;
 else{myArray[myBack]=value;  myBack=newBack;
        } }

QueueElement Queue::front( ) const
{  if (!empty())      return myArray[myFront];
        cerr<<"*** QUEUE IS EMPTY *** \n";      }
void Queue::removeQ()
{       if (empty())  cout<<"**** QUEUE IS EMPTY
****\n";
        else
myFront=(myFront+1)%QUEUE_CAPACITY;}

void Queue::display() const
{ for (int i=myFront ; i<myBack ; i++)
cout<<myArray[i]<<"  "; }
#endif
```

```
void main() {
        Queue q1;
   cout<<"Queue is empty "<<q1.empty( )<<endl;
        q1.addQ(7);
        q1.addQ(77);
        q1.addQ(5);
        q1.addQ(55);
        cout<<"Queue is empty "<<q1.empty( )<<endl;
        q1.display();
        q1.removeQ();
        q1.removeQ();
        cout<<"\n\n";
        q1.display();
        cout<<"\n\n";
        system("pause");}
```

فيكون الخرج الناتج على الشكل التالي:



تمرين 2:

اكتب برنامج يقوم بإنشاء رتل يحوي الأرقام التالية بالترتيب 1 35 55 ومن ثم قم بطباعة العنصر الأول والعنصر الأخير ثم قم بإفراغ الرتل وطباعة عناصره

الحل:

```
#include <iostream>
#include <queue>

using namespace std;

int main() {
```

إنشاء الرتل

```
  queue<int> num;
```

دفع العناصر إلى الرتل

```
  num.push(1);
  num.push(35);
  num.push(55);
```

طباعة العنصر الأول والعنصر الأخير من الرتل

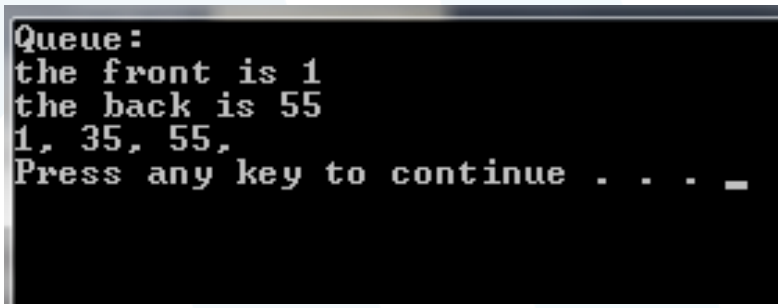
```
  cout << "Queue: "<<endl;
  cout << "the front is "<< num.front()<<endl;
  cout << "the back is "<< num.back()<<endl;
```

إفراغ الرتل وطباعة عناصره

```
  while(!num.empty()) {

    // print the element
    cout << num.front() << ", ";

    num.pop();
  }
```

```
  cout << endl;
 system("pause");
  return 0;
}
```

فيكون الخرج الناتج بالشكل التالي:



تمرين 3:

بفرض أننا نريد تصميم برنامج لإنجاز تمارين على العمليات الحسابية الأساسية، وبشكل أكثر تحديداً، تمارين على إنجاز عملية الجمع لأعداد صحيحة مولدة بشكل عشوائي. بحيث إذا قام الطالب بالإجابة بشكل صحيح يتم توليد مسألة جديدة، ولكن إذا أجاب بشكل خاطئ فإن المسألة يتم تخزينها بحيث يتم إعادة طرحها لاحقاً في نهاية الجلسة.

المطلوب :

اكتب برنامج بلغة C++ لحل المسألة السابقة.

الحل:

من أجل الحصول على نمذجة أفضل للحل سنقوم بتعريف نمط بيانات يدعى RandomInt كصنف من خلال الملف الرأسي RandomInt.h التالي:

```
#ifndef RANDOMINT
```

```
#define RANDOMINT
#include <iostream<
#include <cstdlib<
#include <cassert<
#include <ctime<
using namespace std؛
class RandomInt
{  public:
RandomInt();
RandomInt(int low, int high);
  RandomInt(int seedValue);
RandomInt(int seedValue, int low, int high);
  void Print(ostream & out) const;
RandomInt Generate؛()
  RandomInt Generate(int low, int high);
operator int؛()
 private:
void Initialize(int low, int high؛(
  void SeededInitialize(int seedValue, int low, int high);
  int NextRandomInt؛()
  int myLowerBound, myUpperBound,  myRandomValue؛
  static bool generatorInitialized؛
```

};

inline RandomInt::RandomInt(){

Initialize(0, RAND_MAX);}

inline RandomInt::RandomInt(int low, int high)

{ assert(0 <= low); assert(low < high);

Initialize(low, high);}

inline RandomInt::RandomInt(int seedValue(

{ assert(seedValue >= 0);

SeededInitialize(seedValue, 0, RAND_MAX);}

inline RandomInt::RandomInt(int seedValue, int low, int high)

{ assert(seedValue >= 0); assert(0 <= low); assert(low < high؛(

SeededInitialize(seedValue, low, high);}

inline void RandomInt::Print(ostream & out) const

{ out << myRandomValue{؛

inline ostream & operator<< (ostream & out, const RandomInt & randInt(

{ randInt.Print(out); return out{؛

inline int RandomInt::NextRandomInt()

{ return myLowerBound + (rand() % (myUpperBound - myLowerBound + 1));}

inline RandomInt RandomInt::Generate()

```
{  myRandomValue = NextRandomInt();  return *this;}
inline RandomInt::operator int(){
return myRandomValue;}
#endif
```

يمكن تعريف التوابع الأعضاء في الصنف من خلال الملف RandomInt.cpp كما يلي:

```
# include "stdafx.h"
#include "RandomInt.h"
using namespace std;
bool RandomInt::generatorInitialized = false;
void RandomInt::Initialize(int low, int high)
{  myLowerBound = low;  myUpperBound = high;
  if (!generatorInitialized)
    {  srand(long(time(0)));  generatorInitialized = true;}
  myRandomValue = NextRandomInt;()
{
void RandomInt::SeededInitialize(int seedValue, int low,
int high(
{ myLowerBound = low;
myUpperBound = high;  srand(seedValue);
  generatorInitialized = true;
myRandomValue = NextRandomInt;()
{
```

```
RandomInt RandomInt::Generate(int low, int high)
{  assert(0 <= low);  assert(low < high);
myLowerBound = low;
myUpperBound = high;
  myRandomValue = NextRandomInt();
return *this;}
```

الملف AdditionProblem.h:

```
#include "RandomInt.h"
#include <iostream>
using namespace std;
#ifndef ADDITION_PROBLEM
#define ADDITION_PROBLEM
using namespace std;
class AdditionProblem
{/***** Function Members *****/
public: void Get(int maxAddend)
{myAddend1.Generate(0,maxAddend);
myAddend2.Generate(0, maxAddend);}
void Display(ostream & out){
cout << myAddend1 << " + " << myAddend2 << " = ? ";}
int Answer(){  return (myAddend1 + myAddend2);}
bool Correct(int userAnswer)
```

```
{  return (userAnswer == Answer());}
/***** Data Members *****/
 private:    RandomInt myAddend1, myAddend2;
};
#endif
```

الملف driver.cpp للاختبار:

```
#include "stdafx.h"
#include "AdditionProblem.h"
#include <queue>
#include <iostream>
using namespace std;
void main()
{  int numProblems, maxAddend;
cout << "How many problems would you like? ";
cin >> numProblems;
cout << "What's the largest addend you would like? ";
cin >> maxAddend;

  queue<AdditionProblem>     wrongQueue;
```

```
  AdditionProblem problem;
int userAnswer;
  for (int count = 1; count <= numProblems; count++)
  {    problem.Get(maxAddend);
problem.Display(cout);
 cin >> userAnswer;
    if (problem.Correct(userAnswer))
cout << "Correct!\n\n";
    else    {    cout << "Sorry -- Try again later\n\n";
wrongQueue.push(problem);    }
  }
cout << "\nIf you got any problems wrong, you will now
be given"
          "\na second chance to answer them correctly.\n";
  int wrong = 0;
  while (!wrongQueue.empty())
  {    problem = wrongQueue.front();    wrongQueue.pop();
problem.Display(cout);
    cin >> userAnswer;
    if (problem.Correct(userAnswer))
cout << "Correct!\n\n";
    else
    {
```

```
    cout << "Sorry -- correct answer is "<< problem.Answer() << "\n\n";
    wrong++;
   }
  }
  cout << "\nYou answered " << wrong << " problem"
    << (wrong > 1 ? "s ": " " ) << "incorrectly\n";}
```

فيكون الخرج بالشكل التالي:

```
What's the largest addend you would like? 10
4 + 2 = ? 6
Correct!

2 + 4 = ? 7
Sorry -- Try again later

2 + 8 = ? 8
Sorry -- Try again later

7 + 0 = ? 7
Correct!


If you got any problems wrong, you will now be given
a second chance to answer them correctly.
2 + 4 = ? 6
Correct!

2 + 8 = ? 3
Sorry -- correct answer is 10


You answered 1 problem incorrectly
Press any key to continue . . .
```