**NỘI DUNG THỰC HÀNH**

**Cơ bản**

1. Sử dụng danh sách liên kết để cài đặt cấu trúc Stack, Queue.
2. Sử dụng các hàm PushStack, PopStack, EnQueue, DeQueue để cài đặt.
   1. Về Stack: Trong hàm main, thực hiện việc thêm vào 3 giá trị do người dùng nhập vào (thực hiện 3 lệnh thêm phần tử vào stack), sau đó thực hiện 4 lần lệnh lấy giá trị phần tử ra khỏi stack, nếu có, in giá trị phần tử ra màn hình, nếu không có (stack rỗng), in ra màn hình “STACK RONG, KHONG LAY DUOC PHAN TU”.
   2. Về Queue: Trong hàm main, thực hiện việc thêm vào 3 giá trị do người dùng nhập vào (thực hiện 3 lần lệnh thêm phần tử vào queue), sau đó thực hiện 4 lần lệnh lấy giá trị phần tử ra khỏi queue, nếu có, in giá trị phần tử ra màn hình, nếu không có (queue rỗng), in ra màn hình “QUEUE RONG, KHONG LAY DUOC PHAN TU”.

|  |
| --- |
| //STACK.h  #pragma once  #include "linkedlist.h"  using namespace std;  class STACK  {  private: int nNum;  linkedlist\* Linkedlist;  public:  STACK();  virtual ~STACK();  void push(int);  int pop();  void printStack();  };  //STACK.cpp  #include <iostream>  #include "STACK.h"  #include "linkedlist.h"  #include "element.h"  using namespace std;  STACK::STACK()  {  this->nNum = 0;  this->Linkedlist = new linkedlist();  }  STACK::~STACK()  {  }  void STACK::push(int x)  {  element\* p = new element(x);  this->Linkedlist->InsertFirst(p);  }  int STACK::pop()  {  if (this->Linkedlist->Gethead() == nullptr) {  cout << "STACK RONG, KHONG LAY DUOC PHAN TU";  return NULL;  }  int p = this->Linkedlist->Gethead()->Getdata();  this->Linkedlist->DeleteFirst();  return p;  }  void STACK::printStack()  {  Linkedlist->Travel();  return;  }  //QUEUE.h  #pragma once  #include "linkedlist.h"  using namespace std;  class QUEUE  {  private:  int nNum;  linkedlist\* Linkedlist;  public:  QUEUE();  virtual ~QUEUE();  void push(int);  int pop();  void printQueue();  };  //QUEUE.cpp  #include "QUEUE.h"  #include <iostream>  QUEUE::QUEUE()  {  this->nNum = 0;  this->Linkedlist = new linkedlist();  }  QUEUE::~QUEUE()  {  }  void QUEUE::push(int x)  {  element\* p = new element(x);  this->Linkedlist->InsertFirst(p);  }  int QUEUE::pop()  {  if (this->Linkedlist->Gethead() == nullptr) {  cout << "QUEUE RONG, KHONG LAY DUOC PHAN TU";  return NULL;  }  int p = this->Linkedlist->Gettail()->Getdata();  this->Linkedlist->DeleteTail();  return p;  }  void QUEUE::printQueue()  {  Linkedlist->Travel();  return;  }  // STACK\_QUEUE.cpp : This file contains the 'main' function. Program execution begins and ends there.  #include <iostream>  #include "element.h"  #include "linkedlist.h"  #include "STACK.h"  #include "QUEUE.h"  using namespace std;  int main()  {  cout << "\n-----STACK----\n";  STACK\* s = new STACK();  s->push(7);  s->push(10);  s->push(18);  s->printStack();  int tmp;  cout << "\n";  while (tmp = s->pop()) {  cout << tmp << " ";  }    cout << "\n-----QUEUE----\n";  QUEUE\* q = new QUEUE();  q->push(5);  q->push(7);  q->push(6);  q->push(1);  q->printQueue();  cout << "\n";  while (tmp = q->pop()) {  cout << tmp << " ";  }  } |

**Áp dụng – Nâng cao**

1. Sử dụng hàm InsertTail và DeleteTail trong LinkedList để có phiên bản cài đặt Stack (thêm phần tử vào cuối danh sách và lấy phần tử ở cuối danh sách liên kết) cũng như áp dụng 1 phiên bản khác khi cài đặt Queue (thêm phần tử vào cuối danh sách liên kết và lấy phần tử ở đầu danh sách liên kết).

|  |
| --- |
| //STACK.cpp  . . .  void STACK::push(int x)  {  element\* p = new element(x);  this->Linkedlist->InsertTail(p);  }  int STACK::pop()  {  if (this->Linkedlist->Gettail() == nullptr) {  cout << "STACK RONG, KHONG LAY DUOC PHAN TU";  return NULL;  }  int p = this->Linkedlist-> Gettail()->Getdata();  this->Linkedlist->DeleteTail();  return p;  }  **. . .**  //QUEUE.cpp  . . .  void QUEUE::push(int x)  {  element\* p = new element(x);  this->Linkedlist-> InsertTail (p);  }  int QUEUE::pop()  {  if (this->Linkedlist->Gethead() == nullptr) {  cout << "QUEUE RONG, KHONG LAY DUOC PHAN TU";  return NULL;  }  int p = this->Linkedlist->Gethead()->Getdata();  this->Linkedlist->DeleteHead();  return p;  }  **. . .** |

1. Nhận xét cách cài đặt trên ở phần 1 (áp dụng – nâng cao) so với chương trình mẫu đối với trường hợp stack cũng như queue.

***Về cơ bản thì hai cách cài đặt đều như nhau, chỉ khác là đảo ngược head và tail.***

1. Sử dụng cấu trúc Stack để chuyển giá trị từ cơ số 10 sang cơ số 2.

Gợi ý : thực hiện việc chia liên tiếp giá trị trong cơ số 10 cho 2, lấy phần dư đưa vào stack, cho đến khi giá trị đem đi chia là 0. In giá trị trong stack ra (đó chính là kết quả khi chuyển số từ hệ cơ số 10 sang hệ cơ số 2).

|  |
| --- |
| void DemicalToBinary(int x) {  STACK\* b = new STACK();  while (x != 0) {  b->push(x%2);  x /= 2;  }  cout << "Binary : ";  b->printStack();  return;  } |

**BÀI TẬP ỨNG DỤNG**

**Bài 1**: Tìm đường trong mê cung (thực hiện loang theo chiều rộng <sử dụng queue> hoặc loang theo chiều sâu <sử dụng stack>).

Bài toán: cho ma trận mxn, mỗi phần tử là số 0 hoặc 1.

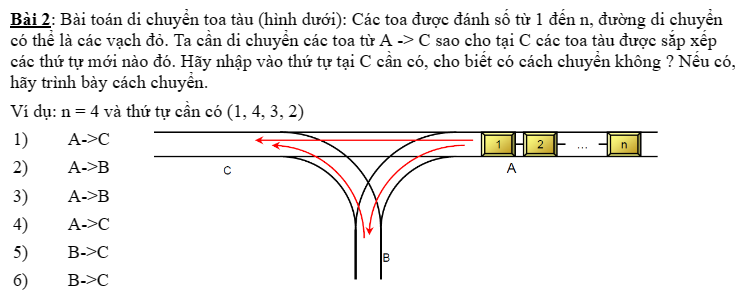
Giá trị 1 : có thể đi tới và giá trị 0 : không thể đi tới được.

Câu hỏi:

Từ ô ban đầu có tọa độ (x1, y1) có thể đi tới ô (x2, y2) không?

Biết rằng từ 1 ô (x,y) chỉ có thể đi qua ô có chung cạnh với ô đang đứng và mang giá trị là 1, ngược lại không có đường đi.

|  |
| --- |
| void Mecung() {  int a[3][3] = {  {1, 1, 0},  {0, 0, 1},  {0, 1, 1}  };  int x1, x2, y1, y2;  x1 = y1 = 0;  x2 = y2 = 2;  if (a[x2][y2] == 0) {  cout << "KHONG THE DI DEN!\n";  return;  }  QUEUE\* qx = new QUEUE();  QUEUE\* qy = new QUEUE();  bool flag = false;  while (a[x1][y1] != 0 && flag == false) {  cout << x1 << " " << y1 << " " << a[x1][y1] << "\n";  if (x1 == x2 && y1 == y2) {  flag = true;  break;  }  a[x1][y1] = 0;  if (x1 + 1 < 3) {  if (a[x1 + 1][y1] != 0) {  qx->push(x1 + 1);  qy->push(y1);  }  }  if (x1 - 1 >= 0) {  if (a[x1 - 1][y1] != 0) {  qx->push(x1 - 1);  qy->push(y1);  }  }  if (y1 + 1 < 3) {  if (a[x1][y1 + 1] != 0) {  qx->push(x1);  qy->push(y1 + 1);  }  }  if (y1 - 1 >= 0) {  if (a[x1][y1 - 1] != 0) {  qx->push(x1);  qy->push(y1 - 1);  }  }  x1 = qx->pop();  y1 = qy->pop();  }  if (flag == true) {  cout << "CO THE DI DEN!\n";  return;  }  else {  cout << "KHONG THE DI DEN!\n";  return;  }  } |



|  |
| --- |
| void toatau1() {  int n;  cout << "Nhap so luong toa tau: ";  cin >> n;  vector <int> a(n);  STACK\* A = new STACK();  STACK\* B = new STACK();  vector <string> C;  cout << "Nhap thu tu toa tau: ";  for (int i = 0; i < n; i++) cin >> a[i];  for (int i = n; i >= 1; i--) A->push(i);  int index = 0;  int tmp;  for (int i = 0; i < n; i++) {  int tmp = A->pop();  if (tmp == NULL || a[i] < tmp) {  tmp = B->pop();  if (tmp == NULL || tmp != a[i]) {  cout << "KHONG THE DI CHUYEN\n";  return;  }  C.push\_back("B-->C");  continue;  }  else {  while (tmp != a[i]) {  B->push(tmp);  C.push\_back("A-->B");  tmp = A->pop();  }  if (tmp == NULL) {  cout << "KHONG THE DI CHUYEN\n";  return;  }  else {  C.push\_back("A-->C");  continue;  }  }  }  for (int i = 0; i < C.size(); i++) {  cout << i + 1 << ").\t" << C[i] << "\n";  }  } |