TRƯỜNG ĐẠI HỌC GIAO THÔNG VÂN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

---------------o0o---------------



**BÀI TẬP LỚN MÔN HỌC**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**CHỦ ĐỀ SỐ 26**

|  |  |  |
| --- | --- | --- |
| Giảng viên hướng dẫn | : | Phạm Đình Phong |
| Sinh viên thực hiện | : | Nhữ Đimhg Đức |
| Lớp | : | CNTT6 K62 |
| Mã sinh viên | : | 211202522 |

*Hà Nội, tháng 11 năm 2022.*

*Hà Nội, tháng 11 năm 2022*

**Hà Nội, tháng 11 năm 2022**

Mục lục

**A . Đề bài**

**Bài 26**

1. Cài đặt cấu trúc dữ liệu trừu tượng vector có bộ lặp xuôi và ngược

2. Cài đặt class câu hỏi trắc nghiệm mỗi câu hỏi gồm câu hỏi dạng văn bản và 4 phương án trong đó có 1 phương án đúng

3. Cài đặt class thi dùng cấu trúc vector đã xây dựng để đọc đữ liệu từ câu hỏi từ file văn bản lưu vào vector. Cho phép tạo ra đề thi gồm k câu hỏi ngẫu nhiên trong vector các câu hỏi, khi hiển thị sáo trộn ví trí các phương án trả lời cho phép người thi và tính điểm.

**B. Bài 1**

***1. Sourses code***

//#ifndef VECTOR\_H\_

#define VECTOR\_H\_

#include<algorithm>

#include<cassert>

#include<iostream>

template <class T>

class reverse\_iterator

{

T \*curr;

public:

reverse\_iterator(T \*c=0) {curr=c;}

reverse\_iterator<T> &operator=(reverse\_iterator<T> it)

{

this->curr=it.curr;

return \*this;

}

reverse\_iterator<T> operator++()//++curr

{

curr--;

return curr;

}

reverse\_iterator<T> operator++(int)//++curr

{

reverse\_iterator<T> tmp=curr;

curr--;

return tmp;

}

T &operator\*() {return \*curr;}

bool operator!=(reverse\_iterator<T> t) {return curr!=t.curr;}

};

template<class T>

class Vector {

private:

unsigned int mySize; //so phan tu vector

unsigned int myCapacity; // kich thuoc bo nho

T \* buffer; //mang chua cac phan tu vector

public:

typedef T \* iterator;

iterator begin() { return buffer; } //vi tri bat dau bo lap xuoi

iterator end() { return buffer + mySize; } // vtri ket thuc bo lap xuoi

typedef reverse\_iterator<T> reverse\_it;

reverse\_it rbegin() {return reverse\_it(buffer+mySize-1);} // vi tri bat dau bo lap nguoc

reverse\_it rend() {return reverse\_it(buffer-1);} // vi tri ket thuc bo lap nguoc

Vector() : buffer(0), mySize(0), myCapacity(0) {}

~Vector() { //ham huy

delete [ ] buffer; //xoa bo nho dem

}

T& back() const { // tra ve gia tri cuoi

return buffer[mySize-1];

}

T& front() const { // tra ve gia tri dau

return buffer[0];

}

int capacity() { //Ham lay dung luong bo nho dem

return myCapacity;

}

unsigned int size() { //Ham lay so phan tu cua vector

return mySize;

}

bool empty() { //Ham kiem tra vector co rong hay khong

return mySize == 0;

}

void pop\_back() { //Xoa phan tu o cuoi vector

mySize--;

}

void reserve(unsigned int); // mo rong vector

void push\_back(T); // them vao cuoi

T &operator [](unsigned int index) { // nap chong toan tu []

return buffer[index];

}

Vector &operator=(Vector<T> V) //Nap chong toan tu = (gan hai vector)

{

this->mySize=V.mySize;

this->myCapacity=V.myCapacity;

if(myCapacity) { //Neu dung luong cua vector > 0

this->buff=new T[myCapacity]; //Khoi tao bo nho chua cac phan tu

for(int i=0;i<mySize;i++) //Sao chep cac phan tu

this->buffer[i]=V.buffer[i];

}

else this->buffer=0;

return \*this;

}

void insert(unsigned int , T); // ham chen vao vi tri

/\*^^virtual\*/ void insert(iterator, T);

void erase(unsigned int,T);

void erase(iterator); // ham xoa qua vi tri bo lap

};

// MEMBER FUNCTIONS

template<class T>

void Vector<T>::reserve(unsigned int newCap) { // mo rong vector

if(buffer == 0) {

mySize = 0;

myCapacity = 0;

}

if(newCap <= myCapacity) { //dung luong moi < dung luong hien tai

return;

}

myCapacity = newCap; // mo rong dung luong

T \* newBuffer = new T[myCapacity]; //Khoi tao bo nho moi lon hon

for(int i=0;i<mySize;i++) //Luu lai cac phan tu da co trong vector

newBuffer[i]=buffer[i];

delete [ ] buffer; ///Xoa bo nho dem cu

buffer = newBuffer; //Tro lai bo nho dem

}

template<class T>

void Vector<T>::push\_back(T value) { //them phan tu vao cuoi vector

if(mySize >= myCapacity) {

reserve(myCapacity\*2); // het dung luowng thi mo rong

}

buffer[mySize++] = value; // them phan tu

}

template<class T>

void Vector<T> ::insert(unsigned int k, T x) { // chen x vao vi tri k

if (myCapacity ==mySize) reserve(myCapacity\*2); // het dung luowng thi mo rong

for (int i=mySize-1;i>=k;i--){ // dich cac phan tu ve cuoi vector

buffer[i+1] = buffer[i];

}

buffer[k]=x; //gan vi tri x vao k

mySize++; //tang so phan tu vector

}

template<class T>

void Vector<T> :: erase(unsigned int k,T o) { // xoa phan tu vi tri k

if(k>mySize-1) return ;

o = buffer[k];

int r=k;

while (r < mySize -1){

buffer[r]=buffer[r+1];

k++;

}

mySize--;

}

int main () {

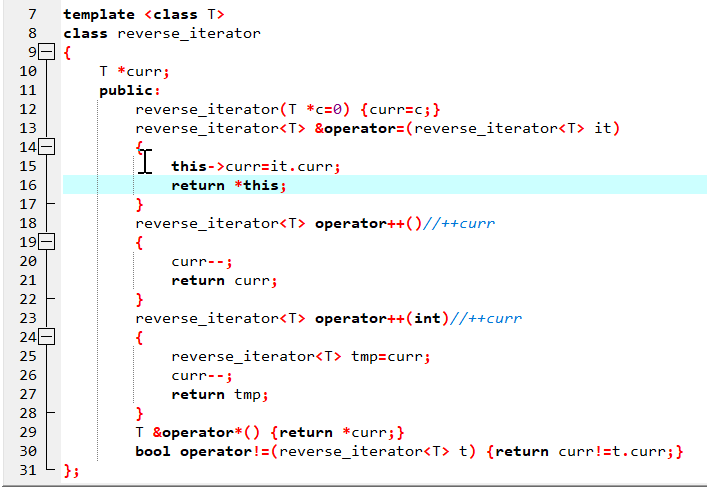
}

***2.Ý tưởng và mô tả thuật toán( thuộc tính ,phương thức các lớp và thời gian chạy phương thức)***

Sử dụng hàm mẫu T để khai báo con trỏ \*curr , xây dựng class reverse\_iterator để thực hiện công viêc khai báo vector có bộ lặp ngược, trong public bao gồm các phương thức:

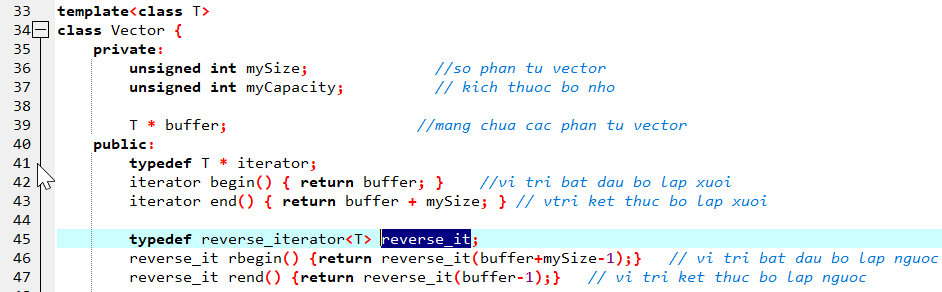
* Hàm khai báo không đối
* Xây dựng toán tử gán =
* Xây dựng toán tử ++curr có đối và không có đối
* Xây đựng toán tử !=
* Xây dựng toán tử trả về iterator \*curr

Cụ thể :



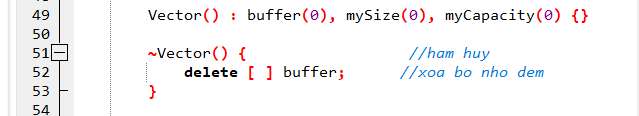
Ứng dụng hầm mẫu xâu dựng class vector có các thuộc tính . phương thức :

* Private : có các thông tin để kiểu số nguyên dương gồm số phần tử vector ,kích thước bộ nhớ và mảng chứa
* Public : khai báo cấu trúc con trỏ \* iterator và reverse\_it , tiến hành khai báo các vị trí đầu và cuối của bộ lặp xuôi ( begin, end) và bộ lặp ngược (rbegin, rend)

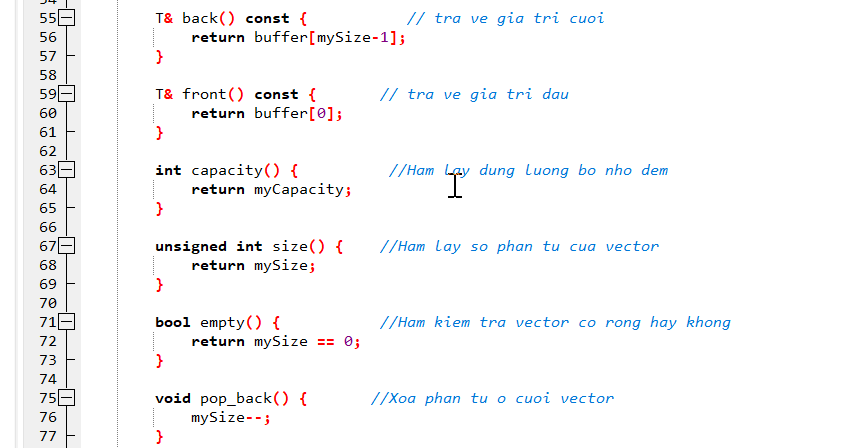


Trong public : xây dựng các phương thức sau :

* Hàm tạo vector không đối
* Hàm hủy vector



* Hàm trả về giá trị đầu front() và cuối back()
* Hàm lấy dung lượng bộ nhớ capacity()
* Hàm trả về số phần tử size()
* Hàm kiểm tra độ rỗng empty()
* Hàm xóa phần tử cuối popback()



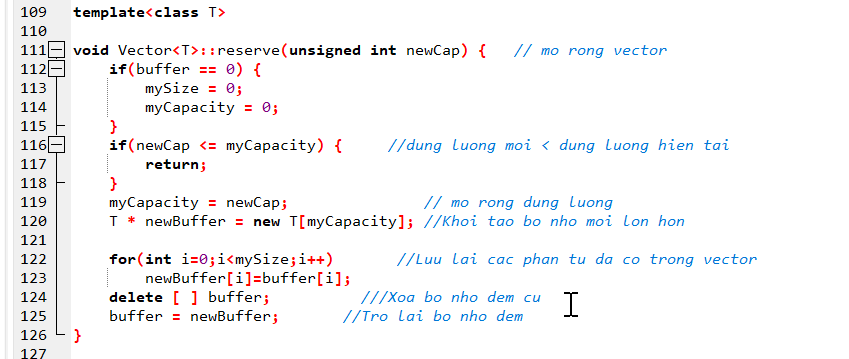
* Hàm mở rộng vector : reserve(unsigned int newCap) : đầu tiên , khởi tạo các thuộc tính private = 0, hàm rỗng khi bộ nhớ mới < bộ nhớ cũ, còn không thì trả gán myCapacity = newCap (cũ gán mới) .

+ Khởi tạo mảng mới lưu có bộ nhớ mới lớn hơn newBuffer

+ Lặp qua các phần tử mảng cũ , lưu nó vào mảng mới, xóa mảng cũ và tạo bộ nhớ đệm cho mảng mới

* Thời gian chạy: dòng 112,113,114 là 1 ; 116 là 2; 117 là 1; 119 là 2; 122 là 2+n ;123 là 2n;124 là 1; 125 là 2

Cụ thể :



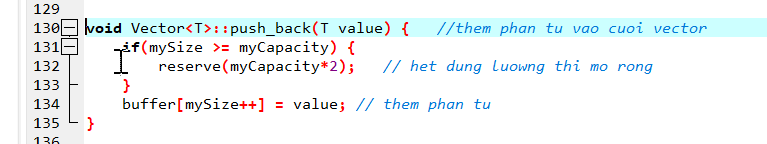
* Hàm thêm phần tử vào cuối push\_back() , khai báo 1 giá trị thêm value

+ Đối chiếu số phần tử vs bộ nhớ , nếu > = thì tăng bộ nhớ gấp đôi

+ Tiến hành thêm

* Xét thời gian chạy : dòng 131 là 2, dòng 132 = reserve + 1; 134 là 2

Cụ thể



* Hàm chèn phần tử insert(unsigned int k, T x) , khai báo giá trị x chèn vào vị trí k

+ Nếu số phần tử bằng bộ nhớ thì mở rộng bộ nhớ