Государственное ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«Донецкий национальный технический университет»

Д09.03.04-ИИ.20.21/5847.ЛР

***Кафедра*** искусственного интеллекта  
 и системного анализа

Лабораторная работа №3

по дисциплине "Объектно ориентированное программирование"

на тему: "Одиночное и множественное наследование. Полиморфизм.”

Проверил:

\_\_\_\_\_\_\_\_\_\_\_\_\_ асс. А.П. Семёнова

(дата, подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ асс. И.В. Савицкая

(дата, подпись)

Выполнил:

\_\_\_\_\_\_\_\_\_\_\_\_\_ ст.гр. ПИ-20г Е.М. Евсеев

(дата, подпись)

Донецк — 2021

Лабораторная работа №3

Тема: «Использование контейнеров, итераторов, алгоритмов библиотеки STL. Реализация функциональных объектов и предикатов.»

Цель: приобрести практические навыки работы с шаблонными функциями, шаблонным классом, стандартной библиотекой шаблонов, изучить определение последовательного и ассоциативного контейнеров, адаптера контейнеров, итератора, адаптера итераторов, функционального объекта, адаптера функций, алгоритмы.

Задание:

1. Контейнеры. Создать контейнер (вектор), добавить в него объекты следующей природы (Верхнетреугольная матрица). Протестировать методы контейнера, скопировать его элементы в другой контейнер.

2. Итераторы. Протестировать работу итераторов различного типа, например, при вводе и просмотре всех элементов контейнера, добавлении, поиске, удалении и пр.

3. Алгоритмы. Протестировать по 2 алгоритма каждой из 4 групп стандартной библиотеки алгоритмов. Указать, где используются стандартные объекты-функции. Использовать функциональный объект из пункта 1.

Листинг программы

#include <iostream>

#include <assert.h>

#include <stdlib.h>

#include <Windows.h>

#include <vector>

#include <iterator>

#include <algorithm>

#include <numeric>

#include <string>

using namespace std;

void cc()

{

cin.clear();

cin.ignore(32767, '\n');

}

class TriMat

{

private:

int Size;

int \*\*M;

public:

TriMat(int n)

{

Size = n;

M = (int \*\*)malloc(Size \* sizeof(int));

for (int k = 0; k < Size; k++)

{

M[k] = (int \*)malloc((Size - k) \* sizeof(int));

for (int j = 0; j <= Size - 1 - k; j++)

{

// cout<<"Введите "<<k+1<<" строку, "<<j+1<< " столбец"<<endl;

// cin >> M[k][j];

M[k][j] = rand() % 15 - 6;

// cc();

};

}

}

// Конструктор скалярной матрицы с заданным элементом на главной диагонали

TriMat(int n, int a)

{

Size = n;

M = (int \*\*)malloc(Size \* sizeof(int));

for (int k = 0; k < Size; k++)

{

M[k] = (int \*)malloc((Size - k) \* sizeof(int));

for (int j = 0; j <= Size - 1 - k; j++)

{

// cout<<"Введите "<<k+1<<" строку, "<<j+1<< " столбец"<<endl;

// cin >> M[k][j];

M[k][j] = rand() % 15 - 6;

// cc();

};

}

for (int k = 0; k < Size; k++)

{

M[k][0] = a;

}

}

void print()

{

cout << endl;

for (int k = 0; k < Size; k++)

{

for (int j = 0; j < k; j++)

cout << "\t";

for (int j = 0; j <= Size - 1 - k; j++)

cout << M[k][j] << "\t";

cout << endl;

}

}

void assign(int i, int j, int a)

{

assert(0 <= i && i < Size && 0 <= j && j < Size);

M[i][j] = a;

}

int el(int i, int j)

{

assert(0 <= i && i < Size && 0 <= j && j < Size);

return M[i][j];

}

int \*operator[](int i)

{

assert(0 <= i && i < Size);

return M[i];

}

TriMat operator+(TriMat B)

{

TriMat C(Size);

for (int k = 0; k < Size; k++)

for (int j = 0; j <= Size - 1 - k; j++)

C[k][j] = M[k][j] + B[k][j];

return C;

}

TriMat operator=(TriMat B)

{

for (int k = 0; k < Size; k++)

for (int j = 0; j <= Size - 1 - k; j++)

M[k][j] = B[k][j];

return \*this;

}

TriMat operator\*(TriMat B)

{

TriMat C(Size);

for (int k = 0; k < Size; k++)

for (int j = k; j < Size; j++)

{

for (int r = k; r <= j; r++)

C[k][j - k] += M[k][r - k] \* B[r][j - r];

}

return C;

}

bool operator<(const TriMat &a) const

{

return (a.M[0][0] % 2 == 0 ? true : false);

}

bool operator+(const TriMat &a) const

{

cout << "was plus" << endl;

return (a.M[0][0] % 2 == 0 ? true : false);

}

bool operator>(const TriMat &a) const

{

return (a.M[0][0] % 2 == 0 ? false : true);

}

bool operator==(const TriMat &a) const

{

return (a.M[0][0] % 2 == 0 ? true : false);

}

bool operator!=(const TriMat &a) const

{

return (a.M[0][0] % 2 == 0 ? false : true);

}

// bool operator<(TriMat){

// return (M[0][0]%2==0?true:false);

// }

friend std::ostream &operator<<(std::ostream &out, const TriMat a);

};

std::ostream &operator<<(ostream &out, TriMat a)

{

// Поскольку operator<< является другом класса Point, то мы имеем прямой доступ к членам Point

a.print();

return out;

}

int main()

{

system("chcp 1251");

system("cls");

TriMat a(6);

a.assign(1, 0, -1);

a[0][1] = -999;

a.print();

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

TriMat b(4, 0);

b.print();

vector<TriMat> elements;

elements.push\_back(a);

elements.push\_back(b);

for (size\_t i = 0; i < 2; i++)

elements.push\_back(TriMat(rand() % 5)); //7 элементов в векторе

vector<TriMat>::iterator it; //инициализация итератора

cout << "\nВыводим элементы вектора:\n\n";

for (it = elements.begin(); it < elements.end(); it++)

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

<< \*it << endl;

cout << "\nВыводим элементы вектора в обратном порядке:\n\n";

for (it = elements.end() - 1; it >= elements.begin(); it--)

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

<< \*it << endl;

cout << "\nВыводим четные элементы вектора:\n\n";

for (it = elements.begin(); it < elements.end(); advance(it, 2))

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

<< \*it << endl;

less<TriMat> y;

cout << "less:" << y(elements[0], elements[1]) << endl;

not\_equal\_to<TriMat> x;

cout << "not\_eq:" << x(elements[0], elements[2]) << endl;

plus<TriMat> z;

z(elements[1], elements[2]);

it = find(elements.begin(), elements.end(), elements[1]);

if (it != elements.end())

std::cout << "Element found in myvector: " << \*it << '\n';

else

std::cout << "Element not found in myvector\n\n";

cout << "count: " << count(elements.begin(), elements.end(), elements[2]) << endl;

cout << "replace " << endl;

replace(elements.begin(), elements.end(), elements[2], elements[0]);

cout << "max " << max(elements[0][0][0], elements[1][0][0]) << endl;

cout << "min " << max(elements[0][0][0], elements[1][0][0]) << endl;

getchar();

return 0;

}

Экранныеформы

Изображение выглядит как текст, черный, экран

Автоматически созданное описание

Рис.1 Отображение верхнетреугольных матриц

Изображение выглядит как текст, снимок экрана, черный, экран

Автоматически созданное описание

Рис.2 Последовательный вывод матриц с помощью итераторов

Изображение выглядит как текст, снимок экрана, черный

Автоматически созданное описание

Рис.3 Обратный вывод матриц с помощью итераторов

Изображение выглядит как текст, экран, черный, снимок экрана

Автоматически созданное описание

Рис. 4 Вывод четных элементов вектора

Изображение выглядит как текст

Автоматически созданное описание

Рис. 5 Алгоритмы стандартной библиотеки

Вывод**:** приобрел практические навыки работы с шаблонными функциями, шаблонным классом, стандартной библиотекой шаблонов, изучил определение последовательного и ассоциативного контейнеров, адаптера контейнеров, итератора,адаптера итераторов, функционального объекта, адаптера функций, алгоритмы.