Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рязанский государственный радиотехнический университет имени

В. Ф. Уткина»

Кафедра «Вычислительная и прикладная математика»

Отчет по лабораторной работе №5

по дисциплине «Объектно-ориентированное программирование»

на тему «Шаблонные классы»

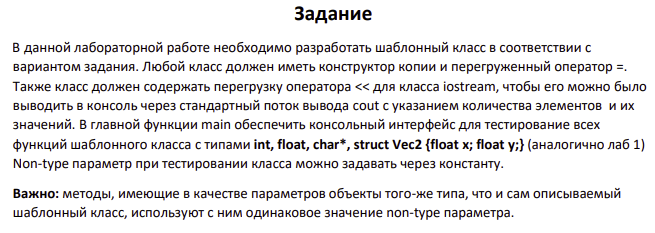
Выполнил: студент гр.1413

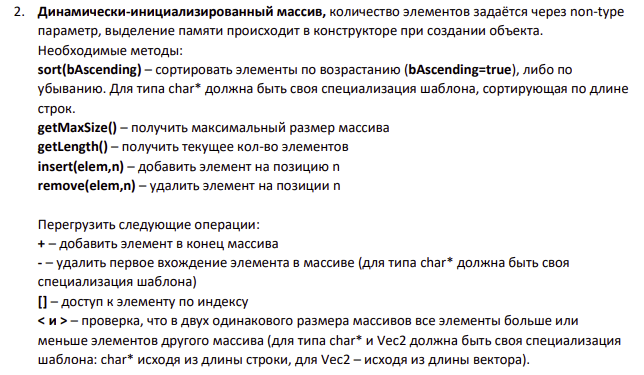
Аверкин М.Е.

Проверил:

доц. Антипов О.В.

**Цель работы**: В данной лабораторной работе необходимо понять принципы работы шаблонного класса и разработать свой шаблонный класс для выполнения поставленной задачи.

**Задание**: 



**Анализ программы:**

Для типов int, float и Vec2 необходимо разработать один шаблон с типом Т и параметром int maxSize, а для типа char\* необходимо создать специализированную версию шаблона, в которой каждый элемент массива char\*(строке).

**Описание структуры:**

template <typename T, int maxSize>

class InitDynArr- общий шаблон класса

Поля:

private:

T\* arr;

int size = 0;

Методы:

InitDynArr () - конструктор по умолчанию

InitDynArr(const InitDynArr& other) - конструктор копии

~InitDynArr()- деструкор

int getSize() const- вывод длины массива

int getMaxSize() const - вывод максимальной возможной длины.

friend ostream& operator<<(ostream& os, InitDynArr<T, maxSize>& arr) – переопределение оператора вывода для работы с классом.

void sort(bool bAscending) – сортировка массива, которая принимает булевую переменную

bAscending, отвечающую за то, будет это сортировка по возрастанию(True) или по убыванию (False)

void insert(const T& elem, int n) –добавление элемента на позицию n

void remove(int n) – удаление элемента с позиции n

T& operator[] – доступ к элементу по индексу

void operator+ - оператор вставки элемента в конец массива

void operator- - оператор удаления первого вхождения элемента

bool operator< - оператор сравнения меньше для класса

bool operator> - оператор сравнения больше для класса

Аналогичными методами обладает и класс

template<int max Size>

class InitDynArr < char\*, maxSize>

**Листинг программы:**

**Interface.h**

#pragma once

#include "DynArray.h"

template <typename T, int maxSize>

void Interface(InitDynArr<T, maxSize> arr) {

int choice;

do {

cout << "Choose on of the functions:" << endl;

cout << "1 - Get maxSize:" << endl;

cout << "2 - Get Size:" << endl;

cout << "3 - Sort:" << endl;

cout << "4 - insert:" << endl;

cout << "5 - Remove:" << endl;

cout << "6 - Print:" << endl;

cout << "7 - Plus:" << endl;

cout << "8 - Minus:" << endl;

cout << "9 - Comparison(<):" << endl;

cout << "10 - Comparison(>):" << endl;

cout << "11 - Get element by index:" << endl;

cout << endl;

cin >> choice;

cout << endl;

switch (choice)

{

case 1: {

cout << "Maximum array size = " << arr.getMaxSize() << endl;

break;

}

case 2: {

cout << "Array size = " << arr.getSize() << endl;

break;

}

case 3: {

bool bAscending;

cout << "Sort ascending (0) or descending (1)" << endl;

cin >> bAscending;

arr.sort(bAscending);

break;

}

case 4: {

T element;

int n;

cout << "Enter the number of the element and element to be added" << endl;

cin >> element >> n;

arr.insert(element, n);

break;

}

case 5: {

int n;

cout << "Enter the number of the element to be removed" << endl;

cin >> n;

arr.remove(n);

break;

}

case 6: {

cout << "----------------------------------------------------------" << endl;

cout << "Values of array:" << arr << endl;

cout << "----------------------------------------------------------" << endl;

break;

}

case 7: {

T element;

cout << "Input element for plus" << endl;

cin >> element;

arr + element;

break;

}

case 8: {

T element;

cout << "Input element for minus" << endl;

cin >> element;

arr - element;

break;

}

case 9: {

T element;

InitDynArr<T, maxSize> arrForTest;

cout << "Input element for Test" << endl;

cin >> element;

arrForTest = arr;

cout << "----------------------------------------------------------" << endl;

cout << "Array(curr):" << arr << endl;

cout << "Array(test):" << arrForTest << endl;

cout << "Comparison result:" << endl;

cout << (arr < arrForTest) << endl;

cout << "----------------------------------------------------------" << endl;

arrForTest + element;

cout << "----------------------------------------------------------" << endl;

cout << "Array(curr):" << arr << endl;

cout << "Array(test):" << arrForTest << endl;

cout << "Comparison result:" << endl;

cout << (arr < arrForTest) << endl;

cout << "----------------------------------------------------------" << endl;

break;

}

case 10: {

T element;

InitDynArr<T, maxSize> arrForTest;

cout << "Input element for Test" << endl;

cin >> element;

arrForTest = arr;

cout << "----------------------------------------------------------" << endl;

cout << "Array(curr):" << arr << endl;

cout << "Array(test):" << arrForTest << endl;

cout << "Comparison result:" << endl;

cout << (arr > arrForTest) << endl;

cout << "----------------------------------------------------------" << endl;

arrForTest + element;

cout << "----------------------------------------------------------" << endl;

cout << "Array(curr):" << arr << endl;

cout << "Array(test):" << arrForTest << endl;

cout << "Comparison result:" << endl;

cout << (arr > arrForTest) << endl;

cout << "----------------------------------------------------------" << endl;

break;

}

case 11: {

int n;

cout << "Enter the element index" << endl;

cin >> n;

cout << "----------------------------------------------------------" << endl;

cout << "Values of array:" << arr[n] << endl;

cout << "----------------------------------------------------------" << endl;

break;

}

default:

cout << "Choice of number from 1 to 11. No other numbers are allowed!" << endl;

break;

}

cout << "Exit - 0" << endl;

cout << "Proceed - 1" << endl;

cin >> choice;

cout << endl;

} while (choice != 0);

}

template <int maxSize>

void chrInterface(InitDynArr<char\*, maxSize> arr) {

int choice;

do {

cout << "Choose on of the functions:" << endl;

cout << "1 - Get maxSize:" << endl;

cout << "2 - Get Size:" << endl;

cout << "3 - Sort:" << endl;

cout << "4 - insert:" << endl;

cout << "5 - Remove:" << endl;

cout << "6 - Print:" << endl;

cout << "7 - Plus:" << endl;

cout << "8 - Minus:" << endl;

cout << "9 - Comparison(<):" << endl;

cout << "10 - Comparison(>):" << endl;

cout << "11 - Get element by index:" << endl;

cout << endl;

cin >> choice;

cout << endl;

switch (choice)

{

case 1: {

cout << "Maximum array size = " << arr.getMaxSize() << endl;

break;

}

case 2: {

cout << "Array size = " << arr.getSize() << endl;

break;

}

case 3: {

bool bAscending;

cout << "Sort ascending (0) or descending (1)" << endl;

cin >> bAscending;

arr.sort(bAscending);

break;

}

case 4: {

char element[256];

int n;

cout << "Enter the number of the element and element to be added" << endl;

cin >> element >> n;

arr.insert(element, n);

break;

}

case 5: {

int n;

cout << "Enter the number of the element to be removed" << endl;

cin >> n;

arr.remove(n);

break;

}

case 6: {

cout << "----------------------------------------------------------" << endl;

cout << "Values of array:" << arr << endl;

cout << "----------------------------------------------------------" << endl;

break;

}

case 7: {

char element[256];

cout << "Input element for plus" << endl;

cin >> element;

arr + element;

break;

}

case 8: {

char element[256];

cout << "Input element for minus" << endl;

cin >> element;

arr - element;

break;

}

case 9: {

char element[256] = "endOfArrayForTest";

InitDynArr<char\*, maxSize> arrForTest;

arrForTest = arr;

cout << "----------------------------------------------------------" << endl;

cout << "Array(curr):" << arr << endl;

cout << "Array(test):" << arrForTest << endl;

cout << "Comparison result:" << endl;

cout << (arr < arrForTest) << endl;

cout << "----------------------------------------------------------" << endl;

arrForTest + element;

cout << "----------------------------------------------------------" << endl;

cout << "Array(curr):" << arr << endl;

cout << "Array(test):" << arrForTest << endl;

cout << "Comparison result:" << endl;

cout << (arr < arrForTest) << endl;

cout << "----------------------------------------------------------" << endl;

break;

}

case 10: {

char element[256] = "endOfArrayForTest";

InitDynArr<char\*, maxSize> arrForTest;

arrForTest = arr;

cout << "----------------------------------------------------------" << endl;

cout << "Array(curr):" << arr << endl;

cout << "Array(test):" << arrForTest << endl;

cout << "Comparison result:" << endl;

cout << (arr>arrForTest) << endl;

cout << "----------------------------------------------------------" << endl;

arrForTest + element;

cout << "----------------------------------------------------------" << endl;

cout << "Array(curr):" << arr << endl;

cout << "Array(test):" << arrForTest << endl;

cout << "Comparison result:" << endl;

cout << (arr > arrForTest) << endl;

cout << "----------------------------------------------------------" << endl;

break;

}

case 11: {

int n;

cout << "Enter the element index" << endl;

cin >> n;

cout << "----------------------------------------------------------" << endl;

cout << "Values of array:" << arr[n] << endl;

cout << "----------------------------------------------------------" << endl;

break;

}

default:

cout << "Choice of number from 1 to 11. No other numbers are allowed!" << endl;

break;

}

cout << "Exit - 0" << endl;

cout << "Proceed - 1" << endl;

cin >> choice;

cout << endl;

} while (choice != 0);

}

**DynArray.h**

#pragma once

#include <iostream>

#include <stdbool.h>

#include <typeinfo>

#include <ctime>

#include <cstring>

using namespace std;

struct Vec2 {

float x = 0.0f,y = 0.0f;

Vec2() = default;

Vec2(float x, float y):

x(x),y(y){

}

friend ostream& operator<<(ostream& os, const Vec2& vec) {

os << vec.x << "-x " << vec.y << "-y; ";

return os;

}

friend istream& operator>>(istream& is, Vec2& vec) {

cout << "x:" << endl;

is >> vec.x;

cout << "y:" << endl;

is >> vec.y;

return is;

}

float getLength() const {

return pow(pow(this->x, 2) + pow(this->y, 2), 0.5);

}

bool operator<(const Vec2& other) const {

return this->getLength() < other.getLength();

}

bool operator>(const Vec2& other) const {

return this->getLength() > other.getLength();

}

bool operator==(const Vec2& other) const {

return this->getLength() == other.getLength();

}

};

template <typename T, int maxSize>

class InitDynArr {

private:

T\* arr;

int size = 0;

public:

InitDynArr() {

size = 0;

arr = new T[maxSize];

}

InitDynArr(const InitDynArr& other) {

size = other.size;

arr = new T[maxSize];

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

{

arr[i] = other.arr[i];

}

}

InitDynArr<T, maxSize>& operator=(const InitDynArr& other){

if (this != &other){

delete[] arr;

size = other.size;

arr = new T[maxSize];

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

arr[i] = other.arr[i];

}

}

return \*this;

}

~InitDynArr() {

delete[] arr;

}

int getSize() const {

return size;

}

int getMaxSize() const {

return maxSize;

}

friend ostream& operator<<(ostream& os, InitDynArr<T, maxSize>& arr) {

for (int i = 0; i < arr.getSize(); ++i)

os << arr[i] << " ";

return os;

}

void sort(bool bAscending);

void insert(const T& elem, int n);

void remove(int n);

T& operator[](int index);

void operator+(const T& elem);

void operator-(const T& elem);

bool operator<(const InitDynArr& other) const;

bool operator>(const InitDynArr& other) const;

};

template <typename T, int maxSize>

void InitDynArr<T, maxSize>::sort(bool bAscending) {

if (bAscending) {

for (int i = 0; i < size - 1; i++) {

int min = i;

for (int j = i + 1; j < size; j++)

if (arr[j] < arr[min])

min = j;

T temp = arr[i];

arr[i] = arr[min];

arr[min] = temp;

}

}

else {

for (int i = size; i >= 1; i--) {

int min = i;

for (int j = i - 1; j >= 0; j--)

if (arr[j] < arr[min])

min = j;

T temp = arr[i];

arr[i] = arr[min];

arr[min] = temp;

}

}

}

template <typename T, int maxSize>

void InitDynArr<T, maxSize>::insert(const T& elem, int n) {

if (n > size){

cout << "You are trying to add an element to a position that is outside the array" << endl;

return;

}

else {

T\* tmp = new T[size + 1];

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

tmp[i] = arr[i];

}

delete[] arr;

arr = tmp;

}

for (int i = size; i > n; i--) {

arr[i] = arr[i - 1];

}

arr[n] = elem;

size++;

}

template <typename T, int maxSize>

void InitDynArr<T, maxSize>::remove(int n) {

if (n > size) {

cout << "You are trying to remove an element from a position that is outside the array" << endl;

return;

}

else {

T\* tmp = new T[size-1];

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

tmp[i] = arr[i];

}

tmp[n] = arr[n + 1];

for (int i = n + 1; i < size - 1; i++) {

tmp[i] = arr[i + 1];

}

size--;

delete[] arr;

arr = tmp;

}

}

template <typename T, int maxSize>

T& InitDynArr<T, maxSize>::operator[](int index) {

if (index > size) {

cout << "Index outside array" << endl;

}

else {

return arr[index];

}

}

template <typename T, int maxSize>

void InitDynArr<T, maxSize>::operator+(const T& elem) {

insert(elem, size);

}

template <typename T, int maxSize>

void InitDynArr<T, maxSize>::operator-(const T& elem) {

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

if (arr[i] == elem){

remove(i);

return;

}

}

}

template <typename T, int maxSize>

bool InitDynArr<T, maxSize>::operator<(const InitDynArr& other) const{

if (size != other.size) {

return false;

}

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

if (arr[i] > other.arr[i]) {

return false;

}

}

return true;

}

template <typename T, int maxSize>

bool InitDynArr<T, maxSize>::operator>(const InitDynArr& other) const

{

if (size != other.size) {

return false;

}

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

if (arr[i] < other.arr[i]) {

return false;

}

}

return true;

}

template <int maxSize>

class InitDynArr<char\*, maxSize>{

private:

char\*\* arr = nullptr;

int size = 0;

public:

InitDynArr(){

size = 0;

arr = new char\* [maxSize];

}

~InitDynArr(){

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

delete[] arr[i];

}

delete[] arr;

}

InitDynArr(const InitDynArr& other) {

size = other.size;

arr = new char\* [maxSize];

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

arr[i] = new char[strlen(other.arr[i]) + 1];

strcpy\_s(arr[i], strlen(other.arr[i]) + 1, other.arr[i]);

}

}

InitDynArr<char\*, maxSize>& operator=(const InitDynArr& other) {

if (this != &other) {

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

delete arr[i];

}

delete arr;

size = other.size;

arr = new char\* [maxSize];

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

{

arr[i] = new char[strlen(other.arr[i]) + 1];

strcpy\_s(arr[i], strlen(other.arr[i]) + 1, other.arr[i]);

}

}

return \*this;

}

int getSize() const {

return size;

}

int getMaxSize() const {

return maxSize;

}

friend ostream& operator<<(ostream& os, InitDynArr<char\*, maxSize>& arr) {

for (int i = 0; i < arr.getSize(); ++i)

os << arr[i] << " ";

return os;

}

void sort(bool bAscending);

void insert(const char\* elem, int n);

void remove(int n);

char\*& operator[](int nindex);

void operator+(const char\* elem);

void operator-(const char\* elem);

bool operator<(const InitDynArr& other) const;

bool operator>(const InitDynArr& other) const;

};

template <int maxSize>

char\*& InitDynArr<char\*, maxSize>::operator[](int index){

if (index > size) {

cout << "Index outside array" << endl;

}

else {

return arr[index];

}

}

template <int maxSize>

void InitDynArr<char\*, maxSize>::sort(bool bAscending) {

if (bAscending) {

for (int i = 0; i < size - 1; i++) {

int min = i;

for (int j = i + 1; j < size; j++)

if (strcmp(arr[j], arr[min]) > 0)

min = j;

char\* temp = arr[i];

arr[i] = arr[min];

arr[min] = temp;

}

}

else {

for (int i = 0; i < size -1; i++) {

int min = i;

for (int j = i + 1; j < size; j++)

if (strcmp(arr[j],arr[min]) < 0)

min = j;

char\* temp = arr[i];

arr[i] = arr[min];

arr[min] = temp;

}

}

}

template <int maxSize>

void InitDynArr<char\*, maxSize>::insert(const char\* elem, int n) {

if (n > size) {

cout << "You are trying to add an element to a position that is outside the array" << endl;

return;

}

else {

for (int i = size; i > n; i--)

{

delete[] arr[i];

arr[i] = new char[strlen(arr[i - 1]) + 1];

strcpy\_s(arr[i], strlen(arr[i - 1]) + 1, arr[i - 1]);

}

arr[n] = new char[strlen(elem) + 1];

strcpy\_s(arr[n], strlen(elem) + 1, elem);

size++;

}

}

template <int maxSize>

void InitDynArr<char\*, maxSize>::remove(int n) {

if (n > size) {

cout << "You are trying to remove an element from a position that is outside the array" << endl;

return;

}

else {

for (int i = n; i < size - 1; i++) {

delete[] arr[i];

arr[i] = new char[strlen(arr[i + 1]) + 1];

strcpy\_s(arr[i], strlen(arr[i + 1]) + 1, arr[i + 1]);

}

delete[] arr[size-1];

size--;

}

}

template <int maxSize>

void InitDynArr<char\*, maxSize>::operator+(const char\* elem) {

insert(elem, size);

}

template <int maxSize>

void InitDynArr<char\*, maxSize>::operator-(const char\* elem) {

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

if (strcmp(arr[i], elem) == 0){

remove(i);

return;

}

}

}

template <int maxSize>

bool InitDynArr<char\*, maxSize>::operator<(const InitDynArr& other) const{

if (size != other.size) {

return false;

}

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

if (strcmp(arr[i], other.arr[i]) > 0) {

return false;

}

}

return true;

}

template <int maxSize>

bool InitDynArr<char\*, maxSize>::operator>(const InitDynArr& other) const

{

if (size != other.size) {

return false;

}

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

if (strcmp(arr[i], other.arr[i]) < 0) {

return false;

}

}

return true;

}

**Lab-5-2.cpp**

#include <iostream>

#include "DynArray.h"

#include "Interface.h"

const int electedMaxSize = 50;

int main(){

srand(time(NULL));

int facechoice;

cout << "Input type:" << endl;

cout << "1 - int:" << endl;

cout << "2 - double:" << endl;

cout << "3 - char\*:" << endl;

cout << "4 - vec:" << endl;

cout << endl;

cin >> facechoice;

cout << endl;

do {

switch (facechoice) {

case 1: {

InitDynArr<int, electedMaxSize> arr;

Interface(arr);

break;

}

case 2: {

InitDynArr<float, electedMaxSize> arr;

Interface(arr);

break;

}

case 3: {

InitDynArr<char\*, electedMaxSize> arr;

chrInterface(arr);

break;

}

case 4: {

InitDynArr<Vec2, electedMaxSize> arr;

Interface(arr);

break;

}

default:

cout << "choice of number from 1 to 4. no other numbers are allowed!" << endl;

break;

}

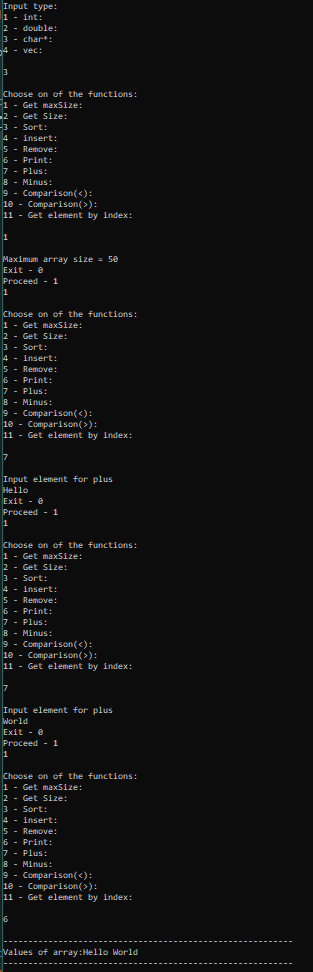
} while (facechoice < 1 or facechoice > 4);

std::cout << "End of programm.\n";

return 0;

}

**Результаты работы программы:**



**Вывод:** Я понял принципы работы шаблонного класса, и разработал свой шаблонный класс для выполнения поставленной задачи.