Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №4

За 4 семестр

По дисциплине «ООТПиСП»

Тема: «Шаблоны функций и классов»

Выполнил: студент 2 курса

Группы ПО-4(2)

Коташевич С.Н.

Проверил: Миндер А.В.

Брест 2021

Лабораторная работа №4

Шаблоны функций и классов

**Цель:** получить практические навыки создания шаблонов и использования их в программах С++.

**Содержание отчета.**

1. Титульный лист: название дисциплины, номер и наименование работы, фамилия, имя, отчество студента, дата выполнения.

2. Постановка задачи.

Следует дать конкретную постановку, т.е указать шаблон какого класса должен быть создан, какие должны быть в нем конструкторы, компоненты-функции, перегруженные операции и т.д.

То же самое следует указать для пользовательского класса.

3. Определение шаблона класса с комментариями.

4. Определение пользовательского класса с комментариями.

5. Реализация конструкторов, деструктора, операции присваивания и операций, которые заданы в варианте задания.

6. То же самое для пользовательского класса.

7. Результаты тестирования. Следует указать для каких типов и какие операции проверены и какие выявлены ошибки (или не выявлены)

**Вариант 13**

13. Класс − однонаправленный список list. Дополнительно перегрузить следующие операции:

+ − добавить элемент в конец (list+item);

-- − удалить элемент из конца (типа list--);

!= − проверка на неравенство.

**Исходный код:**

#include <iostream>

#include <string>

using namespace std;

template <class T> class List {

private:

T\* ptr;

int maxSize;

int currentSize;

public:

List(int maxSize = 10) {

this->currentSize = 0;

this->maxSize = 10;

this->ptr = new T[maxSize];

}

List(const List<T>& lst) {

this->currentSize = lst.currentSize;

this->maxSize = lst.maxSize;

this->ptr = new T[this->maxSize];

for (int i = 0; i < this->currentSize; i++) {

this->ptr[i] = lst.ptr[i];

}

}

~List() {

delete this->ptr;

}

bool isOverflow() {

if (this->currentSize == this->maxSize) {

cout << "Overflow" << endl;

return true;

}

return false;

}

void input(T\* arr, int size) {

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

if (isOverflow()) {

break;

}

this->ptr[i] = arr[i];

}

this->currentSize = size;

}

void print() {

for (int i = 0; i < this->currentSize; i++) {

cout << this->ptr[i] << " ";

}

cout << endl;

}

List& operator=(const List<T>& lst) {

this->currentSize = lst.currentSize;

this->maxSize = lst.maxSize;

for (int i = 0; i < this->currentSize; i++) {

this->ptr[i] = lst.ptr[i];

}

return \*this;

}

List& operator+(const T item) {

if (isOverflow()) {

return \*this;

}

this->ptr[this->currentSize++] = item;

return \*this;

}

List& operator--() {

this->currentSize--;

return \*this;

}

bool operator!=(const List& lst) {

if (this->currentSize != lst.currentSize || this->maxSize != lst.maxSize) {

return true;

}

else {

for (int i = 0; i < this->currentSize; i++) {

if (this->ptr[i] != lst.ptr[i]) {

return true;

}

}

}

return false;

}

};

class Complex {

int real;

int imag;

friend ostream& operator<< (ostream& outstream, const Complex& cpl);

public:

Complex() {

this->real = 0;

this->imag = 0;

}

Complex(int re, int im) {

this->real = re;

this->imag = im;

}

void set\_complex(int re, int im) {

this->real = re;

this->imag = im;

}

string string\_complex() {

string str;

str += to\_string(this->real);

if (this->imag >= 0) {

str += '+';

str += to\_string(this->imag);

str += 'i';

}

else {

str += to\_string(this->imag);

str += 'i';

}

return str;

}

bool operator==(const Complex& cpl) {

return (this->real == cpl.real && this->imag == cpl.imag);

}

bool operator!=(const Complex& cpl) {

return !(\*this == cpl);

}

};

ostream& operator<< (ostream& outstream, Complex& cpl) {

return outstream << cpl.string\_complex();

}

template<class T>

void test(T arr[], int size, T element) {

List<T> lst1;

lst1.input(arr, size);

cout << "List 1: ";

lst1.print();

List<T> lst2;

lst2.input(arr, size/ 2);

cout << "List 2: ";

lst2.print();

lst2 = lst2 + element;

cout << "List 2 + element " << element << ": ";

lst2.print();

cout << "List 1 != List 2: ";

if (lst1 != lst2) {

cout << "True\n";

}

else {

cout << "False\n";

}

cout << "List 1 without the last element: " << endl;

--lst1;

lst1.print();

}

int main()

{

int k[5] = { 1,2,3,4,5 };

char j[5] = { '1','2','3','4','5' };

test<int>(k, 5, 6);

cout << endl;

test<char>(j, 5, '7');

cout << endl;

Complex cm[5];

cm[0].set\_complex(2, 6);

cm[1].set\_complex(7, -6);

cm[2].set\_complex(5, 2);

Complex el;

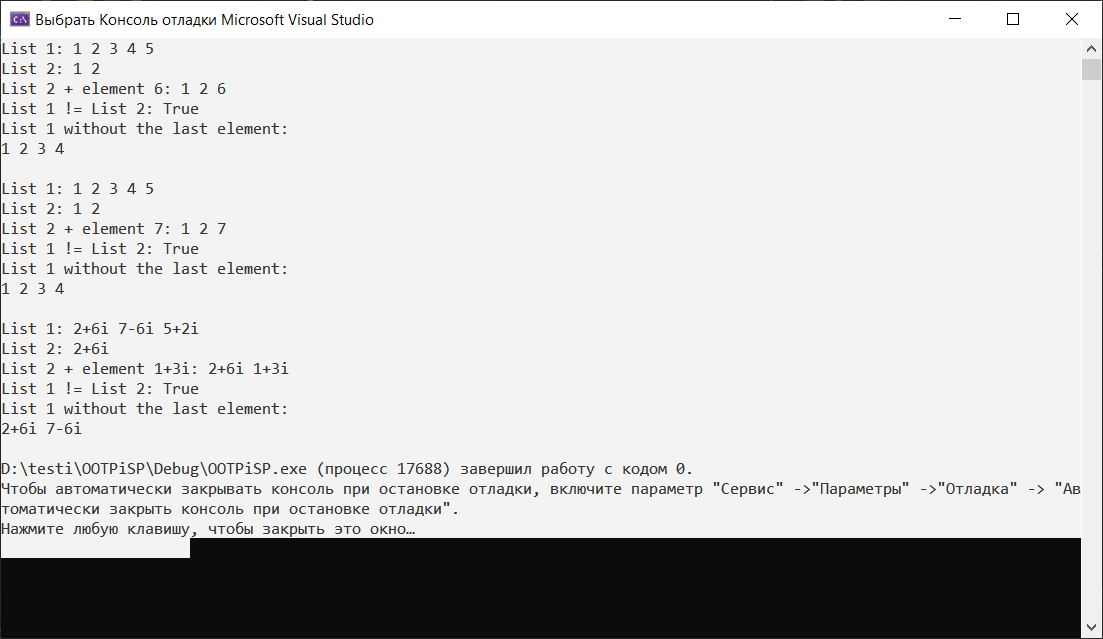
el.set\_complex(1, 3);

test<Complex>(cm, 3, el);

return 0;

}

**Результат работы:**



Вывод: в ходе лабораторной работы получил практические навыки создания шаблонов и использования их в программах С++.