Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

(МТУСИ)



КАФЕДРА: Информационная безопасность

Дисциплина: Программирование в системах информационной безопасности

Лабораторная работа №9

Классы с динамическими структурами данных

Выполнил:

студент группы БСУ1801 Убушуев Б.С

Проверил:

старший преподаватель кафедры ИБ Барков В. В.

Москва 2021

### ЗАДАНИЕ

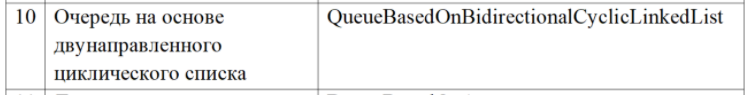
Для типа динамической структуры данных, указанного в индивидуальном задании, разработать соответствующий класс, предусмотрев в нем конструкторы инициализации, копирования, перемещения, деструктор, функции вставки и удаления элемента, просмотра доступного элемента и функцию, проверяющую наличие элементов.

Перегрузить операции присваивания, присваивание с перемещением и потокового вывода для вывода содержимого динамической структуры на экран. Обязательные функции-члены:

|  |  |  |
| --- | --- | --- |
| Стек | Очередь | Дек |
| int GetSize() const;  bool Push(const T &element);  bool Pop(T &element);  bool Peek(T &element); | int GetSize();  bool Push(const T &element);  bool Pop(T &element);  bool Peek(T &element); | int GetSize() const;  bool PushFront(const T &element);  bool PopFront(T &element);  bool PushBack(const T &element);  bool PopBack(T &element);  bool PeekFront(T &element) const;  bool PeekBack(T &element) const; |

**ИНДИВИДУАЛЬНЫЙ ВАРИАНТ ЗАДАНИЯ №25(10)**

Таблица 1 – Индивидуальные варианты заданий.



Листинг 1 – Исходный код файла Complex.cpp

|  |
| --- |
| #include "Complex.h"  double Complex::getReal() const  {  return this->r;  }  double Complex::getImaginary() const  {  return this->phi;  }  void Complex::setR(double r)  {  this->r = r;  }  void Complex::setPhi(double phi)  {  this->phi = phi;  }  Complex::Complex(double r, double phi)  :r(r), phi(phi) { }  ostream& operator << (ostream& stream, const Complex& z)  {  if (z.getImaginary() >= 0)  {  stream << z.getReal() << "+" << z.getImaginary() << "\*i";  }  else  {  stream << z.getReal() << z.getImaginary() << "\*i";  }  return stream;  }  Complex operator\*(Complex a, Complex b)  {  return Complex(a.getReal() \* b.getReal(), a.getImaginary() \* b.getImaginary()); }  Complex operator -(Complex a, Complex b)  {  return Complex(a.getReal() - b.getReal(), a.getImaginary() - b.getImaginary());  }  Complex operator /(Complex a, Complex b)  {  return Complex(((a.getReal() \* b.getReal()) - (a.getImaginary() \* b.getImaginary() \* (-1))) / (pow(b.getReal(), 2) + pow(b.getImaginary(), 2)), ((a.getReal() \* b.getImaginary() \* (-1)) + (a.getImaginary() \* b.getReal())) / (pow(b.getReal(), 2) + pow(b.getImaginary(), 2)));  }  Complex operator +(Complex a, Complex b)  {  return Complex(a.getReal() + b.getReal(), a.getImaginary() + b.getImaginary());  }  Complex pow(const Complex& z, int stepen) {  Complex result(0, 0);  for (int i = 0; i <= stepen; i++)  {  switch (i % 4)  {  case 1:  result.setPhi((factorial(stepen) / (factorial(i) \* factorial(stepen - i))) \* pow(z.getReal(), stepen - i) \* pow(z.getImaginary(), i));  break;  case 2:  result.setR((factorial(stepen) / (factorial(i) \* factorial(stepen - i))) \* pow(z.getReal(), stepen - i) \* pow(z.getImaginary(), i));  break;  case 3:  result.setPhi((factorial(stepen) / (factorial(i) \* factorial(stepen - i))) \* pow(z.getReal(), stepen - i) \* pow(z.getImaginary(), i));  break;  case 0:  result.setR((factorial(stepen) / (factorial(i) \* factorial(stepen - i))) \* pow(z.getReal(), stepen - i) \* pow(z.getImaginary(), i));  break;  }  }  return result;  }  Complex tanh(const Complex& z)  {  return Complex(sinh(2 \* z.getReal()), sin(2 \* z.getImaginary())) / Complex(cosh(2 \* z.getReal()) + cos(2 \* z.getImaginary()), 0);  } |

Листинг 2 – Исходный код файла Complex.h

|  |
| --- |
| #pragma once  #include "iostream"  #include "conio.h"  using namespace std;  class Complex  {  double r, phi;  public:  double getReal() const;  double getImaginary() const;  void setR(double r);  void setPhi(double phi);  Complex(double r, double phi);  friend ostream& operator << (ostream& stream, const Complex& z);  friend Complex operator -(Complex a, Complex b);  friend Complex operator +(Complex a, Complex b);  friend Complex operator /(Complex a, Complex b);  friend Complex operator \*(Complex a, Complex b);  friend Complex pow(const Complex& z, int stepen);  friend Complex tanh(const Complex& z);  };  int factorial(int x); |

Листинг 3 – Исходный код файла main.cpp

|  |
| --- |
| #include "Complex.h"  void main()  {  setlocale(LC\_ALL, "Rus");  double r, phi;  cout << "Введите Действительную и Мнимую часть Комплексного числа:\nr = ";  cin >> r;  cout << "Phi = ";  cin >> phi;  Complex z(r, phi);  cout << "z = " << z;  cout << "\nf(z) = z^3+(1+2i)z^2+(1-2i)z^(-5)= (" << (pow(z, 3)) <<  ") + (" << (Complex(1, 2) \* pow(z, 2)) << ") + ("  << (Complex(1, -2) \* pow(z, -5)) << ") = "  << Complex((pow(z, 3))) + (Complex(1, 2) \* pow(z, 2)) + (Complex(1, -2) \* pow(z, -5));  \_getch();  }  int factorial(int x)  {  if (x == 0) return 1;  return x \* factorial(x - 1);  } |

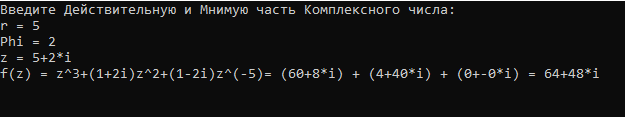


Рисунок 1 – Выполнение lab\_5.exe

**ВЫВОД**

В ходе данной работы были изучены динамические структуры данных, получен навык создания конструкторов копирования, перемещения, деструкторов, перегрузки операций копирования и перемещения.