Дисциплина: «ООП»

Лабораторная работа №1.

**Тема: «Классы»**

Вариант №

Выполнил:

Студент группы !!!!!!!

!!!!!!!!! !.!.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Принял:

Рожков М.А.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Постановка задачи: Цель настоящей работы состоит в ознакомлении студента с правилами организации классов, принятыми при программировании на языке С++. В процессе выполнения настоящей работы каждый студент должен разработать два класса и написать тестовые программы для демонстрации их работоспособности.

**Задание №1**

**Задача 4.**

Необходимо предусмотреть выполнение следующих операций над строками разрабатываемого класса:

-   **int** Length() – определение длины строки,

* void Copy(const String str) – скопировать строку str,
* int Find(char ch, int start) – начиная с индекса start, найти положение символа ch в строке, для которой вызывается метод Find(),
* int FindLast(char ch – найти последнее вхождение символа ch,
* String Substr(int index, int count) – выделение подстроки, начиная с индекса index; count – длина, выделяемой подстроки,
* void Remove(int index, int count) – удаление подстроки, начиная с индекса index; count – длина удаляемой подстроки.
* void Insert(char\* s, int index) - вставка строки в стиле языка C в строку, для которой вызывается метод Insert; index – позиция,  перед которой выполняется вставка.
* void print() – метод для вывода строки на экран дисплея.

 Требования к реализации. Строка должна располагаться в динамической памяти. Поле  для  хранения информации в классе отсутствует.

Код программы:

MyString.h:

#ifndef STRING\_STRUCT\_MYSTRING\_H  
#define STRING\_STRUCT\_MYSTRING\_H  
  
  
class MyString {  
 private:  
 char \*str;  
 public:  
 MyString();  
  
 MyString(const char\* S);  
  
 MyString(const MyString& A);  
  
 MyString(int L);  
  
 ~MyString();  
  
 int Length() const;  
  
 int Find(char ch, int start);  
  
 int Find\_Last(char ch);  
  
 void Print() const;  
  
 void Remove(int index, int count);  
  
 void Insert(const char\* s, int index);  
  
 MyString Substr(int index, int count);  
  
 char operator [](long long i);  
  
 MyString operator =(const MyString &A);  
};  
  
  
#endif //STRING\_STRUCT\_MYSTRING\_H

MyString.cpp:

#include "MyString.h"  
#include <iostream>  
#include <cstring>  
  
MyString::MyString() {  
 str = new char[10];  
 for(int i = 0 ; i < 9 ; i++) {  
 str[i] = 'a';  
 }  
 str[9] = '\0';  
}  
  
MyString::MyString(int L) {  
 str = new char[L + 1];  
 for(int i = 0 ; i < L; i++) {  
 str[i] = 'a';  
 }  
 str[L] = '\0';  
}  
  
MyString::MyString(const char \*S) {  
 str = new char[strlen(S) + 1];  
 for(int i = 0 ; i < strlen(S) ; i++) {  
 str[i] = S[i];  
 }  
 str[strlen(S)] = '\0';  
}  
  
MyString::MyString(const MyString &A) {  
 str =new char[strlen(A.str) + 1];  
 for(int i = 0 ; i < strlen(A.str) ; i++) {  
 str[i] = A.str[i];  
 }  
 str[strlen(A.str)] = '\0';  
}  
  
int MyString::Length() const {  
 return strlen(str);  
}  
  
MyString::~MyString() {  
 delete[] str;  
}  
  
char MyString::operator[](long long i) {  
 return str[i];  
}  
  
void MyString::Print() const {  
 for(int i = 0 ; i < strlen(str) ; i++) {  
 std::cout << str[i];  
 }  
 std::cout <<"\n";  
}  
  
MyString MyString::operator=(const MyString &A) {  
 if(this == &A) {  
 return \*this;  
 }  
 else {  
 delete[] str;  
 str = new char[strlen(A.str) + 1];  
 for (int i = 0; i < strlen(A.str); i++) {  
 str[i] = A.str[i];  
 }  
 str[strlen(A.str)] = '\0';  
 return \*this;  
 }  
}  
  
int MyString::Find(char ch, int start) {  
 int iter = -1;  
 for(int i = start ; i < strlen(str) ; i++) {  
 if(str[i] == ch) {  
 iter = i;  
 break;  
 }  
 }  
 return iter;  
}  
  
int MyString::Find\_Last(char ch) {  
 int iter = -1;  
 for(int i = strlen(str) - 1 ; i >= 0 ; i--) {  
 if(str[i] == ch) {  
 iter = i;  
 break;  
 }  
 }  
 return iter;  
}  
  
MyString MyString::Substr(int index, int count) {  
 MyString A(count);  
 for(int i = index ; i < index + count ; i++) {  
 A.str[i - index] = str[i];  
 }  
 return A;  
}  
  
void MyString::Remove(int index, int count) {  
 char \*A = new char[strlen(str) - count + 1];  
 for(int i = 0 ; i < index ; i++) {  
 A[i] = str[i];  
 }  
 for(int i = index + count ; i < strlen(str) ; i++) {  
 A[i - count] = str[i];  
 }  
 A[strlen(str) - count] = '\0';  
 delete[] str;  
 str = A;  
}  
  
void MyString::Insert(const char \*s, int index) {  
 char \*A = new char[strlen(str) + strlen(s) + 1];  
 for(int i = 0 ; i < index ; i++) {  
 A[i] = str[i];  
 }  
 for(int i = index ; i < index + strlen(s) ; i++) {  
 A[i] = s[i - index];  
 }  
 for(int i = index + strlen(s) ; i < strlen(str) + strlen(s) ; i++) {  
 A[i] = str[i - strlen(s)];  
 }  
 A[strlen(str) + strlen(s)] = '\0';  
 delete[] str;  
 str = A;  
}

Описание функций и полей:

char \*str – поле, в котором хранится строка(адрес начала);

MyString(); -конструктор по умолчанию;

MyString(const char\* S); -конструктор с параметрами;

MyString(const MyString& A); -конструктор копирования;

MyString(int L); -конструктор с параметрами;

~MyString(); -деструткор;

int Length() const; -функция выводящая длину строки;

int Find(char ch, int start); -функция находящая первую букву в строке, начиная с определенного символа;

int Find\_Last(char ch); -функция находящая последнее вхождение

void Print() const; -функция вывода строки;

void Remove(int index, int count); -функция удаления count букв в строке, начиная с index;

void Insert(const char\* s, int index); -функция вставки подстроки(s) в строку, начиная с index;

MyString Substr(int index, int count); -функция выделения подстроки в строке длиной count;

char operator [](long long i); -перегрузка оператора [] для char;

MyString operator =(const MyString &A); -перегрузка оператора присваивания для строки.

Тесты:

main.cpp:

#include <iostream>  
#include "MyString.h"  
  
int main() {  
 MyString Str;  
 std::cout << "Str:";  
 Str.Print();  
 MyString A("ABACABA");  
 std::cout << "A:";  
 A.Print();  
 MyString B(5);  
 std::cout << "B:";  
 B.Print();  
 std::cout << "A length: " << A.Length() << "\n";  
 std::cout << "Position \"A\" from 1 in A: " << A.Find('A', 1) <<"\n";  
 std::cout << "Position of last \"A\" in A: " << A.Find\_Last('A') <<"\n";  
 A.Remove(2, 3);  
 std::cout << "Delete starting from 2 index 3 symbols in A: ";  
 A.Print();  
 A.Insert("ACA", 2);  
 std::cout << "Insert \"ACA\" in A starting from 2 index: ";  
 A.Print();  
 Str = A.Substr(2, 4);  
 std::cout << "Insert in Str substring from A: ";  
 Str.Print();  
 std::cout << "A[1]: " << A[1];  
 return 0;  
}

Результат работы:

Str:aaaaaaaaP

A:ABACABA

B:aaaaa

A length: 7

Position "A" from 1 in A: 2

Position of last "A" in A: 6

Delete starting from 2 index 3 symbols in A: ABBA

Insert "ACA" in A starting from 2 index: ABACABA

Insert in Str substring from A: ACAB

A[1]: B

**Задание №2**

Создание класса комплексного числа. В задаче 16 реализация должна содержать три поля. Первое и втрое поле должны задавать само комплексное число (его действительную и мнимую части), а последнее поле должно содержать модуль комплексного числа.

Разработанный класс (Complex) должен обеспечить выполнение следующих операций:

* сложение,
* вычитание,
* умножение,
* деление,
* вывод комплексного числа на экран дисплея.

Код программы:

Complex.h :

#ifndef COMPLEX\_CLASS\_COMPLEX\_H  
#define COMPLEX\_CLASS\_COMPLEX\_H  
  
  
class Complex {  
private:  
 double a, b;  
 double mod;  
public:  
 Complex(double a1, double b1);  
  
 Complex();  
  
 Complex(const Complex &C);  
  
 void Print() const;  
  
 void Print\_Mod() const;  
  
 Complex operator +(const Complex &c);  
  
 Complex operator -(const Complex &c);  
  
 Complex operator \*(const Complex &c);  
  
 Complex operator /(const Complex &c);  
  
};  
  
  
#endif //COMPLEX\_CLASS\_COMPLEX\_H

Complex.cpp :

#include "Complex.h"  
#include <cmath>  
#include <iostream>  
  
Complex::Complex(double a1, double b1) {  
 a = a1;  
 b = b1;  
 mod = sqrt(a1\*a1 + b1\*b1);  
}  
  
void Complex::Print() const {  
 if (b > 0 || b < 0) {  
 std::cout << a << " + " << b << "i" << "\n";  
 }  
 else if(b == 0) {  
 std::cout << a << "\n";  
 }  
 }  
  
void Complex:: Print\_Mod() const{  
 std::cout<<mod<<"\n";  
 }  
  
Complex::Complex() {  
 a = 0;  
 b = 0;  
 mod = 0;  
}  
  
Complex Complex::operator+(const Complex &c) {  
 Complex a\_new(a + c.a, b + c.b);  
 return a\_new;  
}  
  
Complex Complex::operator-(const Complex &c) {  
 Complex a\_new(a - c.a, b - c.b);  
 return a\_new;  
}  
  
Complex Complex::operator\*(const Complex &c) {  
 Complex a\_new((a\*c.a - b\*c.b),(a\*c.b + c.a\*b));  
 return a\_new;  
}  
  
Complex Complex::operator/(const Complex &c) {  
 if (c.a == 0 && c.b == 0) {  
 std::cout << "Impossible" << "\n";  
 return Complex(0, 0);  
 }  
 else {  
 return Complex (((a \* c.a + b \* c.b) / (c.a \* c.a + c.b \* c.b)), ((b \* c.a - a \* c.b) / (c.a \* c.a + c.b \* c.b)));  
 }  
}  
  
Complex::Complex(const Complex &C) {  
 a = C.a;  
 b = C.b;  
 mod = C.mod;  
}

Описание функций и полей:

double a - поле, в котором храниться вещественная часть комплексного числа;

double b - поле, в котором храниться мнимая часть комплексного числа;

double mod - поле, в котором храниться модуль комплексного числа;

Complex(double a1, double b1); -конструктор с параметрами;

Complex(); -конструктор по умолчанию;

Complex(const Complex &C); -конструктор копирования;

void Print() const; -функция вывода комплексного числа;

void Print\_Mod() const; -функция вывода модуля комплексного числа;

Complex operator +(const Complex &c); -перегрузка оператора сложения для комплексного числа;

Complex operator -(const Complex &c); -перегрузка оператора вычитания для комплексного числа;

Complex operator \*(const Complex &c); -перегрузка оператора умножения для комплексного числа;

Complex operator /(const Complex &c); - перегрузка оператора деления для комплексного числа.

Тесты:

main.cpp:

#include <iostream>  
#include "Complex.h"  
  
int main() {  
 Complex a(1, 2);  
 Complex b;  
 Complex a\_new(a);  
 b = a;  
 b = a + a\_new;  
 std::cout << "Output + :" << "\n";  
 b.Print();  
 b.Print\_Mod();  
 b = a - a\_new;  
 std::cout << "Output - :" << "\n";  
 b.Print();  
 b.Print\_Mod();  
 b = a/a\_new;  
 std::cout << "Output / :" << "\n";  
 b.Print();  
 b.Print\_Mod();  
 b = a\*a\_new;  
 std::cout << "Output \* :" << "\n";  
 b.Print();  
 b.Print\_Mod();  
 return 0;  
}

Результаты работы:

Output + :

2 + 4i

4.47214

Output - :

0

0

Output / :

1

1

Output \* :

-3 + 4i

5

Вывод: В результате проведения работы, мы ознакомились с правилами организации классов, принятыми при программировании на языке С++, а также разработали два класса и написали тестовые программы для демонстрации их работоспособности.