Федеральное агентство связи

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

Факультет информационных технологий и программной инженерии Кафедра: Программная инженерия. Разработка программного обеспечения и приложений искусственного интеллекта в киберфизических системах

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине **«Объектно-ориентированное программирование»**

Тема: Классы

Выполнил: студент 2-го курса группы ИКПИ-42 Терещенко Максим Андреевич
Преподаватель: Петрова Ольга Борисовна

Санкт-Петербург 2025

Постановка задачи

Цель работы – ознакомиться с правилами организации классов в языке C++.

В ходе выполнения работы необходимо:

- Разработать два класса;
- Реализовать все необходимые конструкторы (по умолчанию, с параметрами, копирования);
- Реализовать деструкторы;

Для варианта 5 необходимо реализовать:

- **Задача 5** – класс `String` (строки на основе динамической памяти).
- **Задача 15** – класс `Complex` (комплексные числа, хранение действительной и мнимой частей).

Таблица идентификаторов

Идентификатор	Описание
String()	Конструктор по умолчанию класса `String`
String(const char* str)	Конструктор, создающий строку из C-строки
String(const String& other)	Конструктор копирования
~String()	Деструктор класса `String`
Complex()	Конструктор по умолчанию класса `Complex`
Complex(double r, double i)	Конструктор, задающий действительную и мнимую части
Complex(const Complex& other)	Конструктор копирования
Length() const	Метод, возвращающий длину строки
Copy(const String& str)	Копирование данных из другой строки
Copy(const char* str)	Копирование данных из C-строки
Substr(int index, int count)	Выделение подстроки
Remove(int index, int count)	Удаление части строки
Insert(char* s, int index)	Вставка подстроки в указанную позицию
trim()	Удаление пробелов в начале и конце строки
read()	Ввод строки с консоли
print() const	Вывод строки или комплексного числа
operator+(const Complex& other)	Сложение комплексных чисел
operator-(const Complex& other)	Вычитание комплексных чисел
operator*(const Complex& other)	Умножение комплексных чисел
operator/(const Complex& other)	Деление комплексных чисел

Анализ задачи

Алгоритмическая часть

Класс `String`

Хранит строку в динамической памяти (`char* data`) и её длину (`int length`).

Методы:

- Получение длины (`Length`)
- Копирование строк (`Copy`)
- Выделение подстроки (`Substr`)
- Удаление фрагмента строки (`Remove`)
- Вставка другой строки (`Insert`)
- Удаление пробелов в начале и конце (`trim`)
- Ввод (`read`) и вывод (`print`)

Отчёт по тестированию обработки исключений

Цель

Проверить корректность обработки исключений в классах **String** и **Complex**.
Определить, какие ошибки могут возникнуть при работе программы, и убедиться, что они корректно перехватываются блоками `try-catch` .

Тестовые данные и ожидаемые результаты

1. Тестирование класса `String`

№	Входные данные	Действие	Ожидаемый результат	Обработка исключения
1	`"  Hello World!  "`	Вызов `trim()`	Пробелы в начале и конце удаляются, результат: `Hello World!`	Исключений нет
2	`"Hello World!"` , `Substr(0, 5)`	Получение подстроки	Возвращается строка `"Hello"`	Исключений нет
3	`"Hello World!"` , `Substr(50, 10)`	Попытка выхода за границы строки	`std::out_of_range`	Сообщение: `"Ошибка при получении подстроки: выход за границы"`
4	`"Hello"` , `Insert((char*)" C++", 2)`	Вставка в середину строки	`"He C++llo"`	Исключений нет
5	`"Hello"` , `Insert((char*)"C++", -1)`	Некорректный индекс вставки	`std::invalid_argument`	Сообщение: `"Ошибка при вставке: недопустимый индекс"`
6	`"Hello World!"` , `Remove(0, 6)`	Удаление первых 6 символов	Результат: `"World!"`	Исключений нет

№	Входные данные	Действие	Ожидаемый результат	Обработка исключения
7	`"Hello"` , `Remove(10, 5)`	Удаление за пределами строки	`std::out_of_range`	Сообщение: `"Ошибка при удалении: выход за границы"`

Скриншоты выполнения:

Тест 1:

```
=== String Class Test ===
The original line:    Hello World!
After trim: Hello World!
!!! Error when receiving a substring: Substr: the index is out of range
After Insert: Hello C++ World!
After Remove(0,6): Hello C++ !

=== Complex Class Test ===
c1 + c2 = 0 + 0i
c1 - c2 = 0 + 0i
c1 * c2 = 0 + 0i
!!! Error when working with complex numbers: Division by zero
end of the programm
```

Тест 2:

```
=== String Class Test ===
The original line:    Hello World!
After trim: Hello World!
Substring (0,5): Hello
After Insert: Hello C++ World!
After Remove(0,6): Hello C++ !

=== Complex Class Test ===
c1 + c2 = 0 + 0i
c1 - c2 = 0 + 0i
c1 * c2 = 0 + 0i
!!! Error when working with complex numbers: Division by zero
end of the programm
```

Тест 3:

```
=== String Class Test ===
The original line:    Hello World!
After trim: Hello World!
!!! Error when receiving a substring: Substr: the index is out of range
After Insert: Hello C++ World!
After Remove(0,6): Hello C++ !

=== Complex Class Test ===
c1 + c2 = 0 + 0i
c1 - c2 = 0 + 0i
c1 * c2 = 0 + 0i
!!! Error when working with complex numbers: Division by zero
end of the programm
```

Тест 4:

```
=== String Class Test ===
The original line:    Hello World!
After trim: Hello World!
Substring (0, 5): Hello
After Insert: He C++llo World!
After Remove(0,6): He C++llo !

=== Complex Class Test ===
c1 + c2 = 0 + 0i
c1 - c2 = 0 + 0i
c1 * c2 = 0 + 0i
!!! Error when working with complex numbers: Division by zero
end of the programm
```

Тест 5:

```
=== String Class Test ===
The original line:    Hello World!
After trim: Hello World!
Substring (0, 5): Hello
!!! A common mistake: Insert: The index is out of range
end of the programm
```

Тест 6:

```
=== String Class Test ===
The original line:    Hello World!
After trim: Hello World!
Substring (0, 5): Hello
After Remove(0,6): World!

=== Complex Class Test ===
c1 + c2 = 0 + 0i
c1 - c2 = 0 + 0i
c1 * c2 = 0 + 0i
!!! Error when working with complex numbers: Division by zero
end of the programm
```

Тест 7:

```
=== String Class Test ===
The original line:    Hello World!
After trim: Hello World!
Substring (0, 5): Hello
After Insert:  C++Hello World!
!!! Error when deleting: Remove: the index is out of range

=== Complex Class Test ===
c1 + c2 = 0 + 0i
c1 - c2 = 0 + 0i
c1 * c2 = 0 + 0i
!!! Error when working with complex numbers: Division by zero
end of the programm
```

2. Тестирование класса `Complex`

Для проверки работы класса были проведены два теста: один корректный и один с преднамеренной ошибкой (деление на ноль).

Тест 1. Корректное выполнение

Входные данные:

c1 = (4, 2)

c2 = (1, 1)

Действие:

Выполняется операция деления `c1 / c2` .

Ожидаемый результат:

(3, -1)

Скриншот:

```
=== Complex Class Test ===
c1 + c2 = 5 + 3i
c1 - c2 = 3 + 1i
c1 * c2 = 2 + 6i
c1 / c2 = 3 - 1i
```

Тест 2. Ошибка при делении на ноль

Входные данные:

c1 = (0, 0)
c2 = (0, 0)

Действие:

Выполняется операция деления `c1 / c2`.

Ожидаемый результат:

Выбрасывается исключение `std::runtime_error`.

Фактический результат:

Программа выводит сообщение об ошибке:
Ошибка при работе с комплексными числами: деление на ноль

Скриншот:

```
=== Complex Class Test ===
c1 + c2 = 0 + 0i
c1 - c2 = 0 + 0i
c1 * c2 = 0 + 0i
!!! Error when working with complex numbers: Division by zero
end of the programm
```

Выводы

- В работе были реализованы два класса: `String` и `Complex`.
- Реализованы все необходимые конструкторы и деструкторы.
- Проведено тестирование, показавшее корректность работы методов.

Приложение

В приложение помещаются файлы:

- `String.h`, `String.cpp`
- `Complex.h`, `Complex.cpp`
- `main.cpp`

`String.h`:

```cpp
#ifndef STRING_H
#define STRING_H

#include <iostream>
using namespace std;

class String {
private:
    char* data;
    int length;

public:
    String();
    String(const char* str);
    String(const String& other);
    ~String();

    int Length() const;
    void Copy(const String& str);
    void Copy(const char* str);
    String Substr(int index, int count);
    void Remove(int index, int count);
    void Insert(char* s, int index);
    void trim();
    void read();
    void print() const;
};

#endif
```

String.cpp :

```cpp
#include "String.h"
#include <cstring>
#include <cctype>
#include <stdexcept>

String::String() : data(nullptr), length(0) {}

String::String(const char* str) {
    if (!str) {
        data = nullptr;
        length = 0;
    } else {
        length = strlen(str);
        data = new char[length + 1];
        strcpy(data, str);
    }
}

String::String(const String& other) {
    length = other.length;
    data = new char[length + 1];
    strcpy(data, other.data);
}

String::~String() {
    delete[] data;
}

int String::Length() const {
    return length;
}

void String::Copy(const String& str) {
    delete[] data;
    length = str.length;
    data = new char[length + 1];
    strcpy(data, str.data);
}

void String::Copy(const char* str) {
    delete[] data;
    length = strlen(str);
    data = new char[length + 1];
    strcpy(data, str);
}


String String::Substr(int index, int count) {
    if (index < 0 || index >= length)
        throw std::out_of_range("Substr: the index is out of range");
    if (count <= 0)
        throw std::invalid_argument("Substr: count must be > 0");

    if (index + count > length) count = length - index;

    char* sub = new char[count + 1];
    strncpy(sub, data + index, count);
    sub[count] = '\0';

    String result(sub);
    delete[] sub;
    return result;
}
```

```cpp
void String::Remove(int index, int count) {
    if (index < 0 || index >= length)
        throw std::out_of_range("Remove: the index is out of range");
    if (count <= 0)
        throw std::invalid_argument("Remove: count must be > 0");

    if (index + count > length) count = length - index;

    char* newData = new char[length - count + 1];
    strncpy(newData, data, index);
    strcpy(newData + index, data + index + count);

    delete[] data;
    data = newData;
    length -= count;
}

void String::Insert(char* s, int index) {
    if (!s)
        throw std::invalid_argument("Insert: the row to insert is nullptr");
    if (index < 0 || index > length)
        throw std::out_of_range("Insert: The index is out of range");

    int insertLen = strlen(s);
    char* newData = new char[length + insertLen + 1];

    strncpy(newData, data, index);
    strcpy(newData + index, s);
    strcpy(newData + index + insertLen, data + index);

    delete[] data;
    data = newData;
    length += insertLen;
}


void String::trim() {
    if (!data) return;

    int start = 0;
    while (isspace((unsigned char)data[start])) start++;

    int end = length - 1;
    while (end >= start && isspace((unsigned char)data[end])) end--;

    int newLen = (end - start + 1 > 0) ? end - start + 1 : 0;

    char* newData = new char[newLen + 1];
    strncpy(newData, data + start, newLen);
    newData[newLen] = '\0';

    delete[] data;
    data = newData;
    length = newLen;
}

void String::read() {
    char buffer[1024];
    cin.getline(buffer, 1024);
    Copy(buffer);
}

void String::print() const {
    if (data) cout << data;
}
```

Complex.h：

```
#ifndef COMPLEX_H
#define COMPLEX_H

#include <iostream>
using namespace std;

class Complex {
private:
    double real;
    double imag;

public:
    Complex();
    Complex(double r, double i);
    Complex(const Complex& other);

    Complex operator+(const Complex& other) const;
    Complex operator-(const Complex& other) const;
    Complex operator*(const Complex& other) const;
    Complex operator/(const Complex& other) const;

    void print() const;
};

#endif
```

Complex.cpp：

```cpp
#include "Complex.h"

Complex::Complex() : real(0), imag(0) {}

Complex::Complex(double r, double i) : real(r), imag(i) {}

Complex::Complex(const Complex& other) : real(other.real), imag(other.imag) {}

Complex Complex::operator+(const Complex& other) const {
    return Complex(real + other.real, imag + other.imag);
}

Complex Complex::operator-(const Complex& other) const {
    return Complex(real - other.real, imag - other.imag);
}

Complex Complex::operator*(const Complex& other) const {
    return Complex(real * other.real - imag * other.imag,
                   real * other.imag + imag * other.real);
}

Complex Complex::operator/(const Complex& other) const {
    double denom = other.real * other.real + other.imag * other.imag;
    if (denom == 0.0) {
        throw std::runtime_error("Division by zero");
    }
    return Complex(
        (real * other.real + imag * other.imag) / denom,
        (imag * other.real - real * other.imag) / denom
    );
}

void Complex::print() const {
    cout << real;
    if (imag >= 0) cout << " + " << imag << "i";
    else cout << " - " << -imag << "i";
}
```

main.cpp：

```cpp
#include "String.h"
#include "Complex.h"
#include <iostream>
#include <stdexcept>
using namespace std;

int main() {
    try {
        cout << "=== String Class Test ===" << endl;
        String s1("   Hello World!   ");
        cout << "The original line: ";
        s1.print();
        cout << endl;

        s1.trim();
        cout << "After trim: ";
        s1.print();
        cout << endl;

        try {
            String sub = s1.Substr(0, 5);
            cout << "Substring (0,5): ";
            sub.print();
            cout << endl;
        } catch (const out_of_range& ex) {
            cerr << "Ошибка при получении подстроки: " << ex.what() << endl;
        }

        try {
            s1.Insert((char*)" C++", 5);
            cout << "After Insert: ";
            s1.print();
            cout << endl;
        } catch (const invalid_argument& ex) {
            cerr << "Ошибка при вставке: " << ex.what() << endl;
        }

        try {
            s1.Remove(0, 6);
            cout << "After Remove(0,6): ";
            s1.print();
            cout << endl;
        } catch (const out_of_range& ex) {
            cerr << "Ошибка при удалении: " << ex.what() << endl;
        }

        cout << "\n=== Complex Class Test ===" << endl;
        Complex c1(3, 4), c2(1, -2);

        try {
            Complex sum = c1 + c2;
            cout << "c1 + c2 = "; sum.print(); cout << endl;

            Complex diff = c1 - c2;
            cout << "c1 - c2 = "; diff.print(); cout << endl;

            Complex prod = c1 * c2;
            cout << "c1 * c2 = "; prod.print(); cout << endl;

            Complex quot = c1 / c2;
            cout << "c1 / c2 = "; quot.print(); cout << endl;

        } catch (const runtime_error& ex) {
            cerr << "Ошибка при работе с комплексными числами: " << ex.what() << endl;
```

```
        }

    } catch (const bad_alloc& ex) {
        cerr << "Ошибка выделения памяти: " << ex.what() << endl;
    } catch (const exception& ex) {
        cerr << "Общая ошибка: " << ex.what() << endl;
    }

    return 0;
}
```