Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «УрФУ имени первого Президента России Б.Н.Ельцина»

ИРИТ-РТФ

Центр ускоренного обучения

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ N 6**

по дисциплине «Современные языки программирования»

Студент группы РИВ-210938у: Вахрушев Н.А.

Преподаватель: Архипов Н.А.

Екатеринбург 2023

Цель

Необходимо получить навыки работы с механизмом описания классов C++.

Ход работы

Ниже представлены блок схемы разработанных алгоритмов:

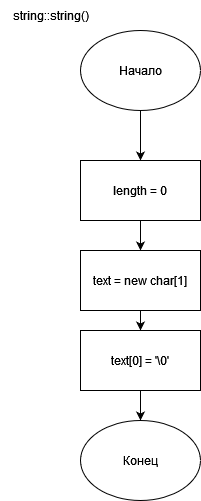


Рисунок 1 – Конструктор

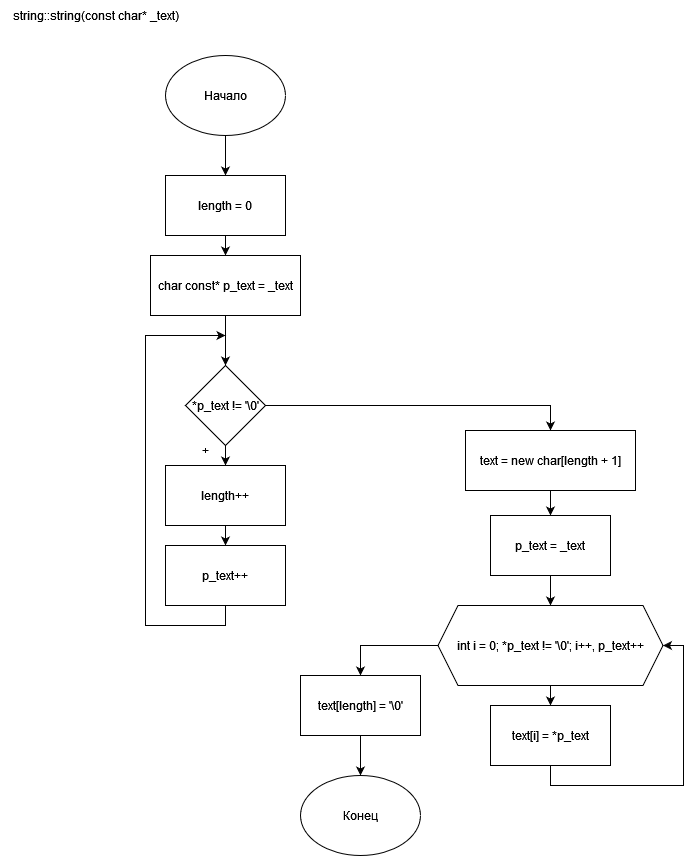


Рисунок 2 –Конструктор

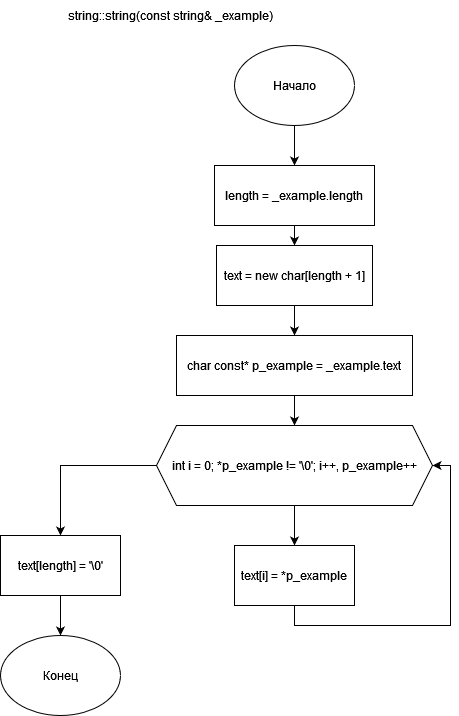


Рисунок 3 – Конструктор

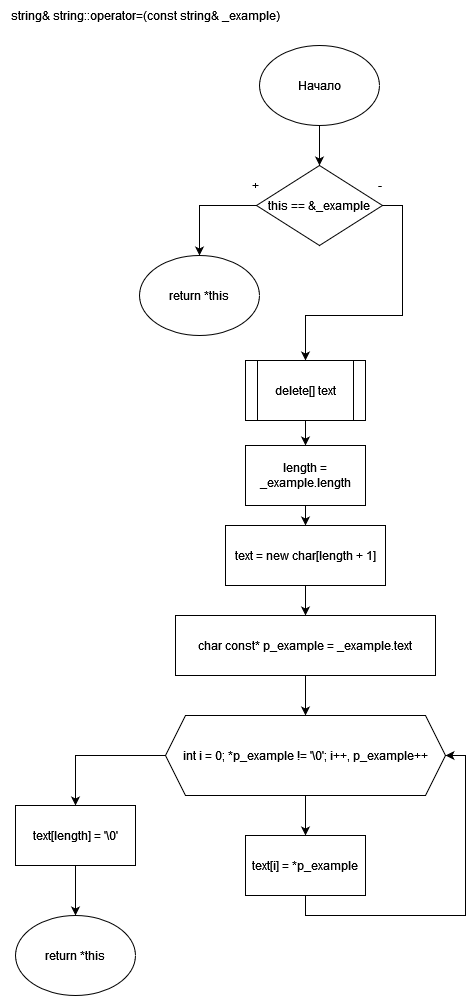


Рисунок 4 - Оператор =

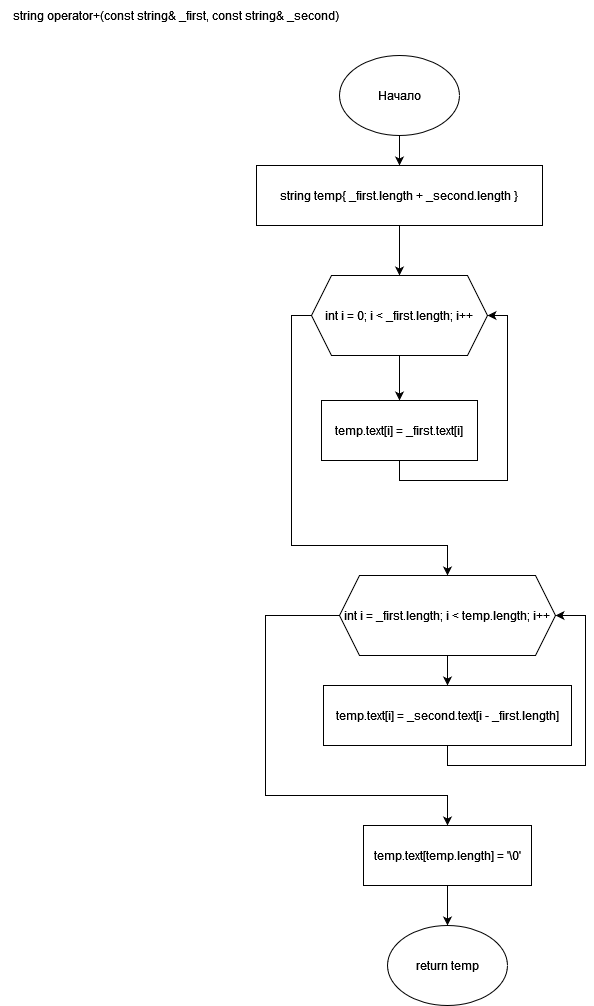


Рисунок 5 – Оператор +

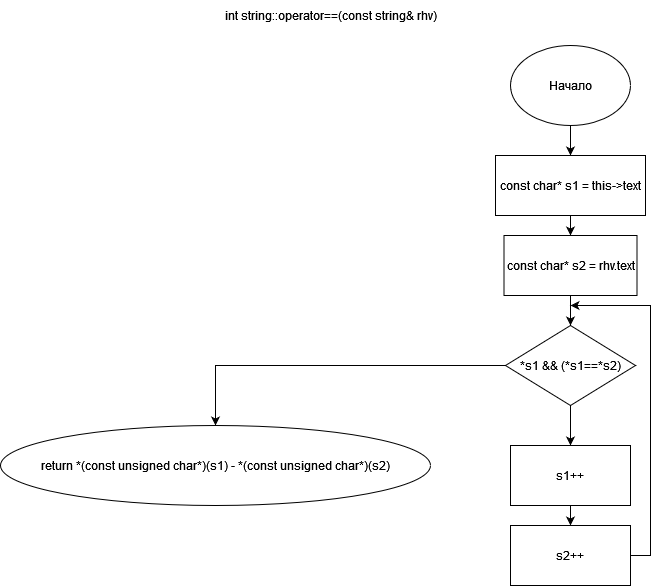


Рисунок 6 - Оператор ==

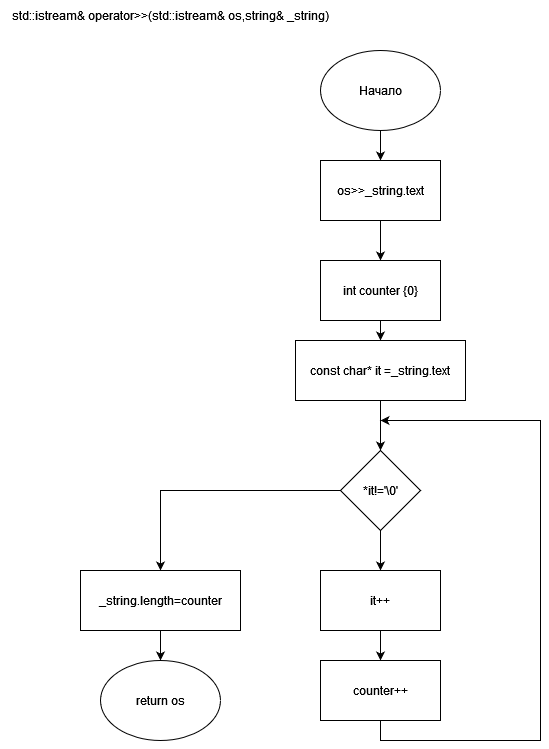


Рисунок 7 - Оператор >>

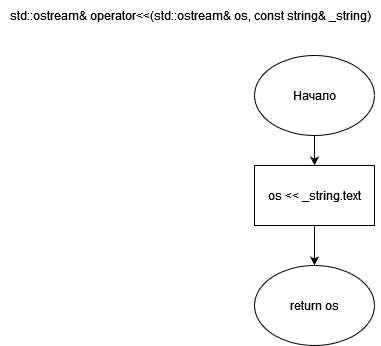


Рисунок 8 - Оператор <<

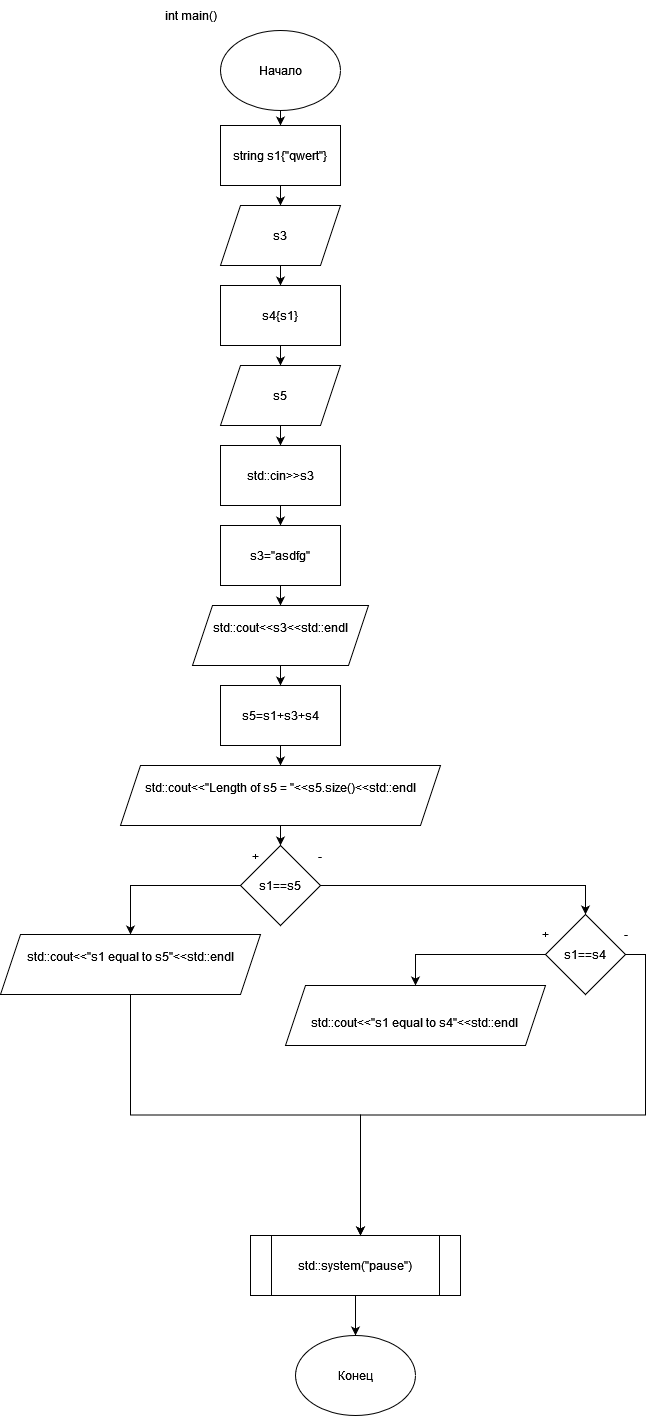


Рисунок 9 - Функция main

Ниже представлены листинги программ:

#include "string.h"

#include <iostream>

int main()

{

string s1{"qwert"},s3,s4{s1},s5;

std::cin>>s3;

s3="asdfg";

std::cout<<s3<<std::endl;

s5=s1+s3+s4;

std::cout<<"Length of s5 = "<<s5.size()<<std::endl;

if(s1==s5)

{

std::cout<<"s1 equal to s5"<<std::endl;

}

else if(s1==s4) std::cout<<"s1 equal to s4"<<std::endl;

std::system("pause");

}

#ifndef STRING\_H

#define STRING\_H

#include <iostream>

class string

{

private:

char\* text; //Указатель на массив типа char, в котором будет храниться строка

int length; //Количество символов в строке, за исключением \0

public:

string(const char\* \_text);

string();

string(const string&);

string(string&&) noexcept;

string(int);

~string() { delete[] text; }

int size() const { return length; }

char& operator[](int i) { return text[i]; }

const char operator[](int i) const { return text[i]; }

string& operator=(const string&);

string& operator=(string&&) noexcept;

string& operator=(const char\*);

int operator==(const string& rhv);

friend string operator+(const string&, const string&);

friend std::ostream& operator<<(std::ostream& os, const string& \_string);

friend std::istream& operator>>(std::istream& os,string& \_string);

};

#endif

#include "string.h"

string::string(string&& example) noexcept: length(example.length)

{

text = example.text;

example.text = nullptr;

example.length = 0;

}

string::string()

{

length = 0;

text = new char[1];

text[0] = '\0';

}

string::string(int i) : length(i)

{

text = new char[i + 1];

for (int i = 0; i <= length; i++) text[i] = '\0';

}

string::string(const char\* \_text)

{

length = 0;

char const\* p\_text = \_text;

while (\*p\_text != '\0') { length++; p\_text++; }

text = new char[length + 1];

p\_text = \_text;

for (int i = 0; \*p\_text != '\0'; i++, p\_text++) text[i] = \*p\_text;

text[length] = '\0';

}

string::string(const string& \_example)

{

length = \_example.length;

text = new char[length + 1];

char const\* p\_example = \_example.text;

for (int i = 0; \*p\_example != '\0'; i++, p\_example++) text[i] = \*p\_example;

text[length] = '\0';

}

std::ostream& operator<<(std::ostream& os, const string& \_string)

{

os << \_string.text;

return os;

}

string& string::operator=(const string& \_example)

{

if (this == &\_example) return \*this;

delete[] text;

length = \_example.length;

text = new char[length + 1];

char const\* p\_example = \_example.text;

for (int i = 0; \*p\_example != '\0'; i++, p\_example++) text[i] = \*p\_example;

text[length] = '\0';

return \*this;

}

string& string::operator=(string&& \_example) noexcept

{

if (this == &\_example) return \*this;

delete[] text;

length = \_example.length;

text = \_example.text;

\_example.text = nullptr;

\_example.length = 0;

return \*this;

}

string& string::operator=(const char\* \_example)

{

length = 0;

delete[] text;

const char\* p\_example = \_example;

while (\*p\_example != '\0') { length++; p\_example++; }

text = new char[length + 1];

for (int i = 0; \*\_example != '\0'; i++, \_example++) text[i] = \*\_example;

text[length] = '\0';

return \*this;

}

string operator+(const string& \_first, const string& \_second)

{

string temp{ \_first.length + \_second.length };

for (int i = 0; i < \_first.length; i++) temp.text[i] = \_first.text[i];

for (int i = \_first.length; i < temp.length; i++) temp.text[i] = \_second.text[i - \_first.length];

temp.text[temp.length] = '\0';

return temp;

}

int string::operator==(const string& rhv)

{

const char\* s1 = this->text;

const char\* s2 = rhv.text;

while(\*s1 && (\*s1==\*s2))

{

s1++;

s2++;

}

return \*(const unsigned char\*)(s1) - \*(const unsigned char\*)(s2);

}

std::istream& operator>>(std::istream& os,string& \_string)

{

os>>\_string.text;

int counter {0};

const char\* it =\_string.text;

while(\*it!='\0')

{

it++;

counter++;

}

\_string.length=counter;

return os;

}

Вывод

В результате работы были приобретены навыки работы с классами в C++.