**Московский государственный технический**

**университет им. Н. Э. Баумана.**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Основы программирования»

Отчет по индивидуальному домашнему заданию

Вариант 21

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-12Б |  | преподаватель каф. ИУ5 |
| Поляков Леонид |  | Правдина А. Н. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2023 г.

**Постановка задачи**

Слова текста из малых латинских букв записаны не менее чем через один пробел; текст оканчивается точкой. БЕЗ ИСПОЛЬЗОВАНИЯ конструкции STRING:

а) написать программу ввода такого текста с клавиатуры;

б) напечатать все слова, отличающиеся от последнего слова, во всех словах текста, оканчивающихся на ov, ova, in, ina, ev, eva, заменить это окончание на idze.

Обработка вводимого текста производится в рантайме, печатающийся текст должен отображаться на экране. Должна быть возможность использования клавиши BACKSPACE для стирания последнего напечатанного символа. Текст вводится до символа точки (‘.’).

**Разработка алгоритма**

Для решения поставленной задачи было принято решение написать собственную реализацию вектора используя шаблоны. Такой вектор позволяет работать с любыми примитивными типами данных, и значительно облегчает работу со строками (массивами типа char), вокруг которой строится данное домашнее задание. Вектор представляет собой обёртку над динамическим массивом, позволяющую работать с ним, как со списком.

**Структура Vector:**

**private поля:**

* T\* vec; // указатель на динамический массив типа T
* size\_t size; // текущий размер динамического массива
* size\_t capasity; // текущая вместимость динамического массива
* const size\_t growthFactor = 2; // коэффициент расширяемости дин. массива

**public поля(методы):**

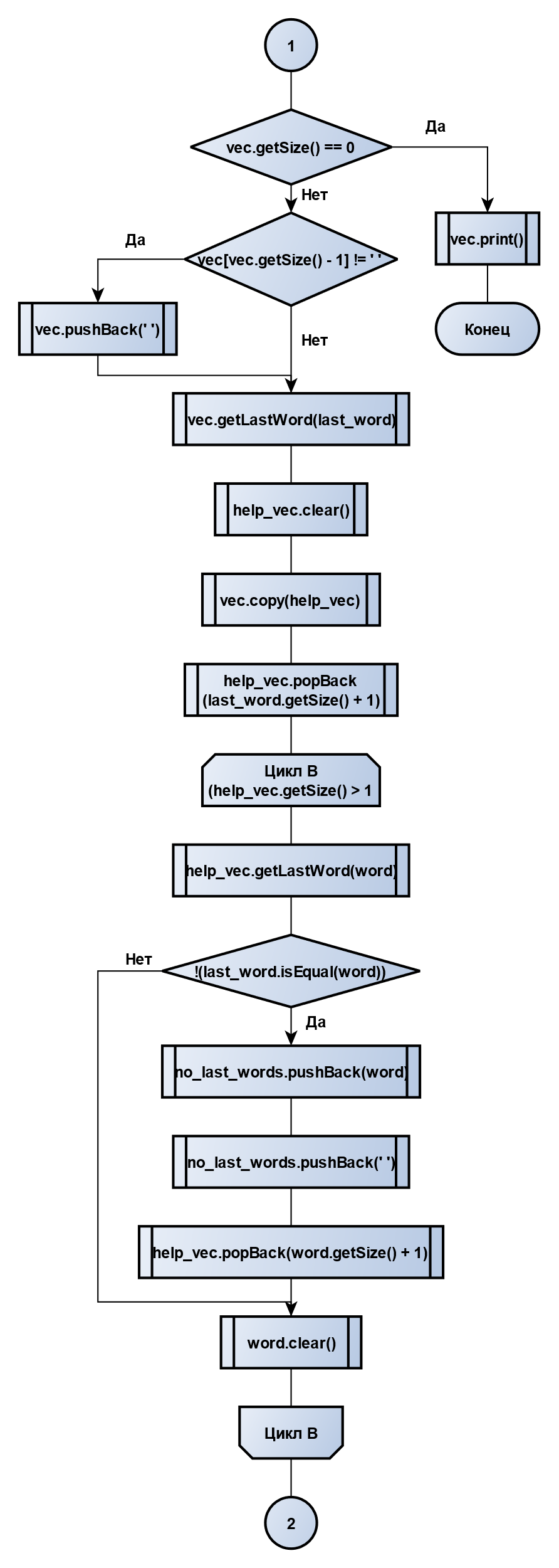
* Vector(); // конструктор структуры Vector
* Vector(size\_t size); // перегрузка конструктора для инициализации непустого вектора (size – размер вектора, который нужно создать)
* ~Vector(); // деструктор структуры Vector
* Vector<T>& operator= (const Vector<T>& v); // перегрузка оператора присваивания (v – вектор, передаваемый в качестве значения для присваивания)
* T& operator[] (const size\_t index); // определение оператора [] для структурных переменных Vector (index – индекс элемента вектора)
* const T& operator[] (const size\_t index) const; // перегрузка определения оператора[] (константная версия, возвращающая ссылку на константное значение)
* void reset(); // метод для очистки памяти, используемой вектором
* void reset(const Vector<T>& v); // перегрузка метода reset(). v – вектор, значение которого получит структурная переменная, вызывающая этот метод
* void clear(); // метод для очистки вектора
* Vector<T>& pushBack(T data); // метод для добавления элемента в конец вектора(data – добавляемый элемент)
* Vector<T>& pushBack(const Vector<T>& v); // перегрузка метода pushBack() для добавления передаваемого вектора в конец вектора, вызывающего данный метод(v – вектор, который будет добавлен в конец)
* Vector<T>& popBack(size\_t pop\_size = 1); // метод для удаления элементов вектора с конца(pop\_size – количество элементов, которые необходимо удалить(по умолчанию – 1))
* Vector<T>& reverse(); // метод для переворачивания вектора
* T getDataByIndex(size\_t index) const; // метод для получения элемента вектора по его индексу(index – индекс элемента)
* size\_t getSize() const; // геттер, возвращающий текущий размер вектора
* size\_t getCapasity() const; // геттер, возвращающий текущую вместимость вектора
* void print() const; // метод для вывода содержимого вектора на экран
* void copy(Vector<T>& v); // метод для копирования вектора(v – вектор, в который будет скопировано значение вектора, вызывающего данный метод)
* bool isEqual(const Vector<T>& v) const; // метод для сравнения векторов. Вернёт true, если векторы равны, иначе вернёт false(v – вектор, который сравнивается с вектором, вызывающим данный метод)
* Vector<T>& slice(Vector<T>& slice, const size\_t begin, const size\_t end); // метод для взятия среза вектора(slice – вектор, в который будет записан срез вектора, вызывающего данный метод, begin – индекс начала среза(включительно), end – индекс конца среза(невключительно))
* Vector<T>& getLastWord(Vector<T>& v) const; // метод для взятия последнего слова из вектора типа char. (v – вектор, в который запишется последнее слово)
* static void fill\_vectors(Vector<T>& ov, Vector<T>& ova, Vector<T>& in, Vector<T>& ina, Vector<T>& ev, Vector<T>& eva, Vector<T>& idze);

// статический метод для заполнения семи переданных векторов данными, которые фигурируют в задаче

**Функция main():**

* vec, last\_word, no\_last\_words, help\_vec, word, ending - векторы для считывания и обработки текста
* ov, ova, ev, eva, in, ina, idze – векторы для хранения служебных данных, фигурирующих в условии задания
* char character – переменная для хранения текущего введённого символа
* unsigned flag – флаг, отвечающий за длину слова

**Считывание и первичная обработка строки**

**Проверяем, что строка, введённая пользователем имеет ненулевой размер, приводим строку к виду с пробелом в конце и записываем в вектор no\_last\_words слова, отличающиеся от последнего**

**Замена окончаний в словах и запись слов в правильном порядке**

**Текст программы**

**Vector.h**

#pragma once

#include<iostream>

namespace Vector {

template <typename T>

struct Vector

{

private:

T\* vec;

size\_t size;

size\_t capasity;

const size\_t growthFactor = 2;

public:

Vector();

Vector(size\_t size);

~Vector();

Vector<T>& operator= (const Vector<T>& v);

T& operator[] (const size\_t index);

const T& operator[] (const size\_t index) const;

void reset();

void reset(const Vector<T>& v);

void clear();

Vector<T>& pushBack(T data);

Vector<T>& pushBack(const Vector<T>& v);

Vector<T>& popBack(size\_t pop\_size = 1);

Vector<T>& reverse();

T getDataByIndex(size\_t index) const;

size\_t getSize() const;

size\_t getCapasity() const;

void print() const;

void copy(Vector<T>& v);

bool isEqual(const Vector<T>& v) const;

Vector<T>& slice(Vector<T>& slice, const size\_t begin, const size\_t end);

Vector<T>& getLastWord(Vector<T>& v) const;

static void fill\_vectors(Vector<T>& ov, Vector<T>& ova, Vector<T>& in, Vector<T>& ina, Vector<T>& ev, Vector<T>& eva, Vector<T>& idze);

};

}

template<typename T>

inline Vector::Vector<T>::Vector() {

vec = new T[1];

capasity = 1;

size = 0;

}

template<typename T>

inline Vector::Vector<T>::Vector(size\_t size) {

vec = new T[size];

capasity = size;

this->size = size;

}

template<typename T>

inline Vector::Vector<T>::~Vector() {

delete[] vec;

vec = nullptr;

}

template<typename T>

inline Vector::Vector<T>& Vector::Vector<T>::operator=(const Vector<T>& v) {

if (this == &v)

return \*this; // Защита от самоприсваивания

this->reset(v);

return \*this;

}

template<typename T>

inline T& Vector::Vector<T>::operator[](const size\_t index) {

if (index >= 0 and index < size)

return vec[index];

std::cout << "Index Error" << std::endl;

}

template<typename T>

inline const T& Vector::Vector<T>::operator[](const size\_t index) const {

if (index >= 0 and index < size)

return vec[index];

std::cout << "Index Error" << std::endl;

}

template<typename T>

inline void Vector::Vector<T>::reset() {

delete[] vec;

vec = nullptr;

}

template<typename T>

inline void Vector::Vector<T>::reset(const Vector<T>& v) {

size = 0;

this->pushBack(v);

}

template<typename T>

inline void Vector::Vector<T>::clear() {

size = 0;

}

template<typename T>

inline Vector::Vector<T>& Vector::Vector<T>::pushBack(T data) {

if (size == capasity) {

T\* temp = new T[capasity \* growthFactor];

for (size\_t i{}; i < size; i++) {

temp[i] = vec[i];

}

delete[] vec;

capasity \*= growthFactor;

vec = temp;

}

vec[size] = data;

size++;

return \*this;

}

template<typename T>

inline Vector::Vector<T>& Vector::Vector<T>::pushBack(const Vector<T>& v) {

if (v.getSize() == 0)

return \*this;

for (size\_t i{}; i < v.getSize(); i++) {

this->pushBack(v[i]);

}

return \*this;

}

template<typename T>

inline Vector::Vector<T>& Vector::Vector<T>::popBack(size\_t pop\_size) {

size -= pop\_size;

return \*this;

}

template<typename T>

inline Vector::Vector<T>& Vector::Vector<T>::reverse() {

size\_t half = static\_cast<size\_t>(size / 2);

for (size\_t i{}; i < half; i++) {

std::swap(vec[i], vec[size - 1 - i]);

}

return \*this;

}

template<typename T>

inline T Vector::Vector<T>::getDataByIndex(size\_t index) const {

if (index >= 0 and index < size)

return vec[index];

std::cout << "Index Error" << std::endl;

}

template<typename T>

inline size\_t Vector::Vector<T>::getSize() const {

return size;

}

template<typename T>

inline size\_t Vector::Vector<T>::getCapasity() const {

return capasity;

}

template<typename T>

inline void Vector::Vector<T>::print() const {

for (size\_t i{}; i < size; i++) {

std::cout << vec[i];

}

std::cout << std::endl;

}

template<typename T>

inline void Vector::Vector<T>::copy(Vector<T>& v) {

if (v.getSize() == 0) {

for (size\_t i{}; i < size; i++) {

v.pushBack(vec[i]);

}

}

else

std::cout << "Vector is not empty" << std::endl;

}

template<typename T>

inline bool Vector::Vector<T>::isEqual(const Vector<T>& v) const {

if (v.getSize() != size)

return false;

size\_t counter{};

for (size\_t i{}; i < size; i++) {

if (vec[i] == v[i])

counter++;

}

if (counter == size)

return true;

return false;

}

template<typename T>

inline Vector::Vector<T>& Vector::Vector<T>::slice(Vector<T>& slice, const size\_t begin, const size\_t end) {

if (slice.getSize() == 0) {

for (size\_t i = begin; i < end; i++) {

slice.pushBack((\*this)[i]);

}

return slice;

}

std::cout << "Vector is not empty" << std::endl;

return \*this;

}

template<typename T>

inline Vector::Vector<T>& Vector::Vector<T>::getLastWord(Vector<T>& v) const {

if (v.getSize() == 0 and this->getSize() > 1) {

for (size\_t elem = this->getSize() - 2; (\*this)[elem] != ' '; elem--) {

v.pushBack((\*this)[elem]);

if (elem == 0)

break;

}

v.reverse();

return v;

}

std::cout << "Source Vector is empty or destination Vector is not empty" << std::endl;

return v;

}

template<typename T>

inline void Vector::Vector<T>::fill\_vectors(Vector<T>& ov, Vector<T>& ova, Vector<T>& in, Vector<T>& ina, Vector<T>& ev, Vector<T>& eva, Vector<T>& idze) {

ov.pushBack('o');

ov.pushBack('v');

ova.pushBack('o');

ova.pushBack('v');

ova.pushBack('a');

in.pushBack('i');

in.pushBack('n');

ina.pushBack('i');

ina.pushBack('n');

ina.pushBack('a');

ev.pushBack('e');

ev.pushBack('v');

eva.pushBack('e');

eva.pushBack('v');

eva.pushBack('a');

idze.pushBack('i');

idze.pushBack('d');

idze.pushBack('z');

idze.pushBack('e');

}

**main.cpp**

#include <conio.h>

#include "Vector.h"

int main() {

Vector::Vector<char> vec, last\_word, no\_last\_words, help\_vec, word, ending, ov, ova, ev, eva, in, ina, idze;

Vector::Vector<char>::fill\_vectors(ov, ova, in, ina, ev, eva, idze);

char character;

unsigned flag{};

//Считываем строку

do {

character = \_getch();

if (!(((97 <= character) and (character <= 122)) or (character == ' ') or (character == '\b') or (character == '.')))

continue;

else if (character == ' ' and (vec.getSize() == 0 or vec[vec.getSize() - 1] == ' ')) {

\_putch(character);

help\_vec.pushBack(character);

continue;

}

else if (character == '\b') {

if (help\_vec.getSize() == 0)

continue;

if (vec.getSize() > 1 and vec[vec.getSize() - 1] == ' ')

vec.popBack();

if (help\_vec[help\_vec.getSize() - 1] != ' ')

vec.popBack();

help\_vec.popBack();

\_putch(character);

\_putch(' ');

\_putch(character);

continue;

}

else if (character == '.') {

\_putch(character);

continue;

}

\_putch(character);

help\_vec.pushBack(character);

vec.pushBack(character);

} while (character != '.');

//Если ничего значащего не ввели

if (vec.getSize() == 0) {

vec.print();

exit(0);

}

//Приводим строку к виду с пробелом в конце

if (vec[vec.getSize() - 1] != ' ')

vec.pushBack(' ');

vec.getLastWord(last\_word); // Получаем последнее слово строки

help\_vec.clear(); // Очищаем help\_vec, чтобы скопировать туда vec

vec.copy(help\_vec); // Копируем вектор vec в help\_vec

help\_vec.popBack(last\_word.getSize() + 1); // Удаляем из вектора help\_vec последнее слово

//Записываем в вектор no\_last\_words слова, отличающиеся от последнего

while (help\_vec.getSize() > 1) {

help\_vec.getLastWord(word);

if (!(last\_word.isEqual(word))) {

no\_last\_words.pushBack(word);

no\_last\_words.pushBack(' ');

}

help\_vec.popBack(word.getSize() + 1);

word.clear();

}

//Записываем слова в том порядке, в котором их вводил пользователь, параллельно проверяя окончания

while (no\_last\_words.getSize() > 1) {

no\_last\_words.getLastWord(word);

switch (word.getSize()) {

case 1:

flag = 1;

help\_vec.pushBack(word);

help\_vec.pushBack(' ');

case 2:

if (flag != 1) {

flag = 2;

if (word.isEqual(ov) or word.isEqual(in) or word.isEqual(ev))

help\_vec.pushBack(idze);

else

help\_vec.pushBack(word);

help\_vec.pushBack(' ');

}

default:

if (flag != 1 and flag != 2) {

word.slice(ending, word.getSize() - 3, word.getSize());

if (ending.isEqual(ova) or ending.isEqual(ina) or ending.isEqual(eva)) {

ending.clear();

word.slice(ending, 0, word.getSize() - 3);

help\_vec.pushBack(ending);

help\_vec.pushBack(idze);

help\_vec.pushBack(' ');

}

else {

ending.clear();

word.slice(ending, word.getSize() - 2, word.getSize());

if (ending.isEqual(ov) or ending.isEqual(in) or ending.isEqual(ev)) {

ending.clear();

word.slice(ending, 0, word.getSize() - 2);

help\_vec.pushBack(ending);

help\_vec.pushBack(idze);

}

else

help\_vec.pushBack(word);

help\_vec.pushBack(' ');

}

}

no\_last\_words.popBack(word.getSize() + 1);

word.clear();

ending.clear();

flag = 0;

break;

}

}

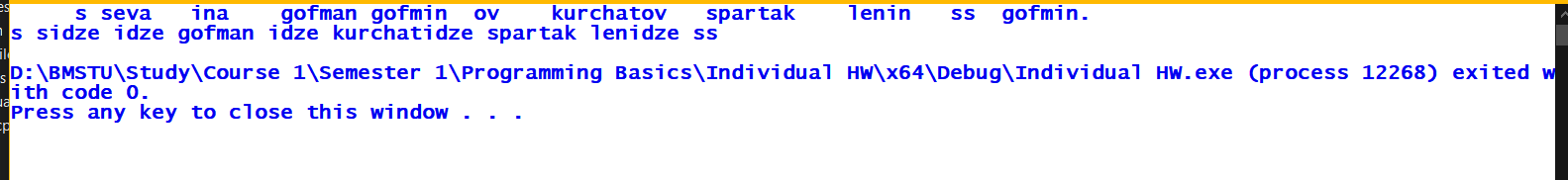
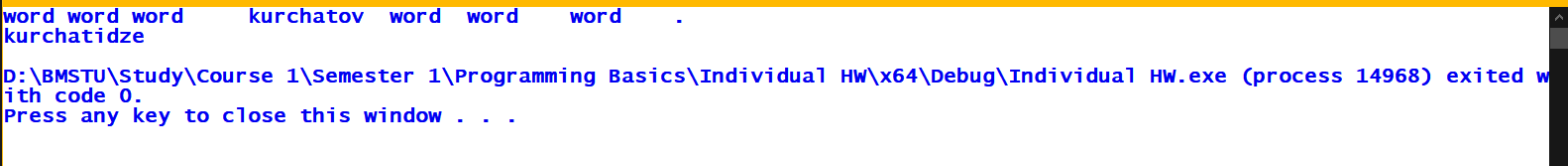
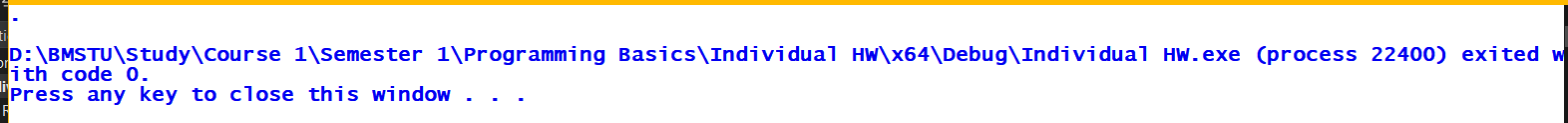
std::cout << std::endl;

help\_vec.print();

return 0;

}

**Анализ результатов**

****

**Результат работы программы соответствует ожиданиям и полностью удовлетворяет требованиям ТЗ**

**Использованные источники**

* **Веб-сайт кафедры ИУ5, курс «основы программирования»**

[**https://cpp1.wiki.iu5edu.ru**](https://cpp1.wiki.iu5edu.ru)

* **Веб-сайт кафедры ИУ5 «автоматизированные системы обработки информации и управления»**

**https://e-learning.bmstu.ru/iu5/mod/folder/view.php?id=277**