Балаковский инженерно-технологический институт - филиал

федерального государственного автономного образовательного учреждения

высшего образования

«Национальный исследовательский ядерный университет «МИФИ»

Факультет атомной энергетики и технологий

Кафедра «Информатика и управление в технических системах»

ПРАКТИЧЕСКАЯ РАБОТА №2

по дисциплине

«Языки программирования»

Вариант №9

Выполнил: студент группы ИФСТ-2з

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Никольский А.Г.

« 02 » декабря 2020г.

|  |  |
| --- | --- |
|  | Проверил ст. преподаватель каф. ИУС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Михеев И.В.  «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_\_г. |

Балаково 2020

Цель работы: изучение алгоритмов ветвления, технологии применения условного оператора и оператора варианта в языке программирования СИ.

Задание №1.

Окно выполнения программы изображено на рис. 1.

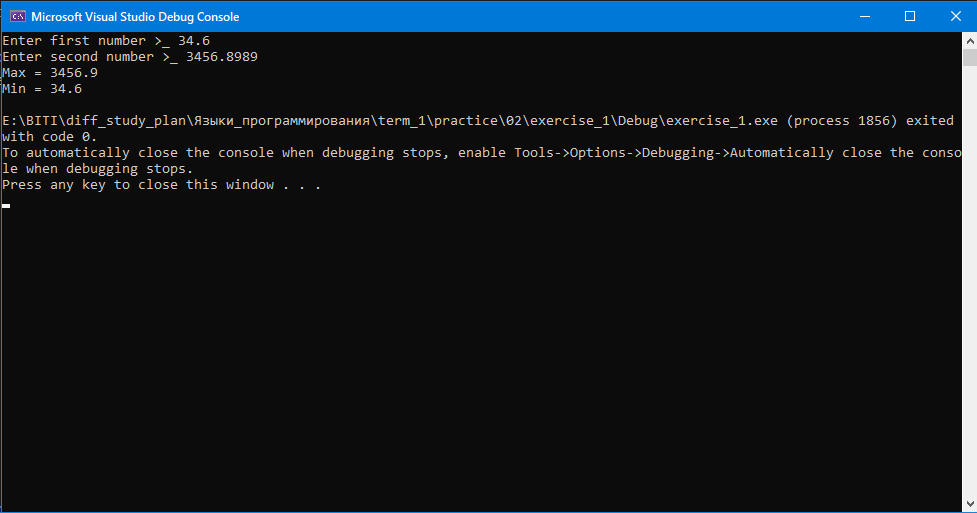


Рисунок 1 - Окно выполнение программы задания №1

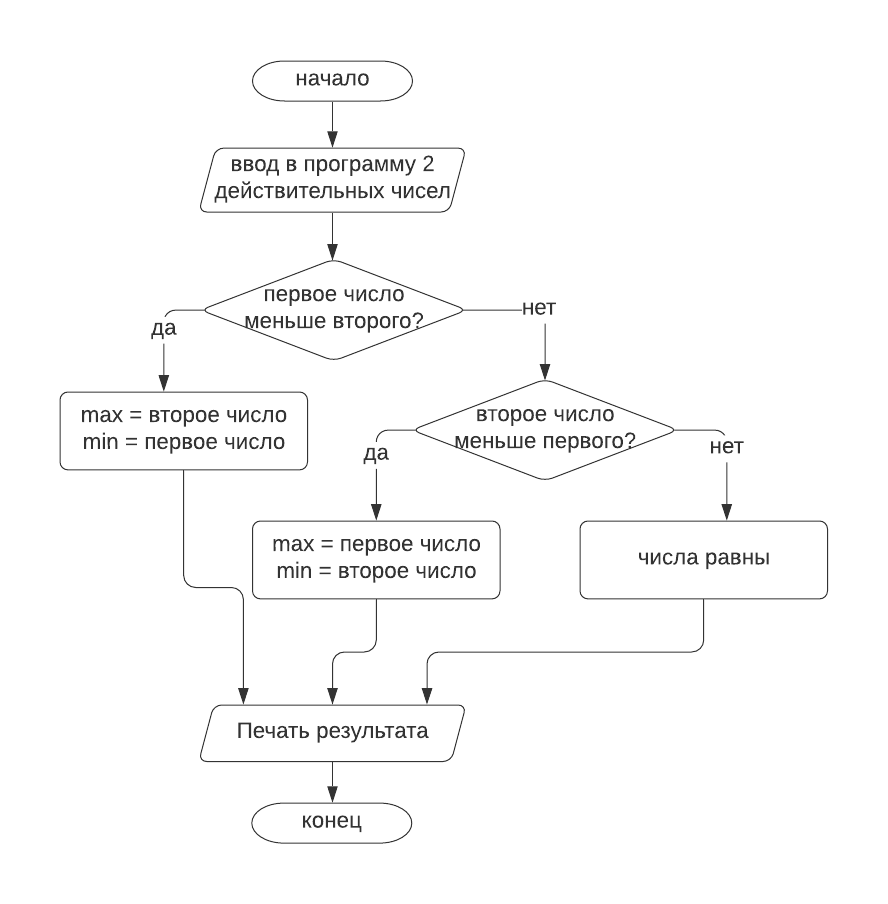
Блок схема алгоритма изображена на рис.2. 

Рисунок 2 - Блок схема к заданию №1

Исходный текст программы:

// created by a.nikolsky 02.12.2020

#include <iostream>

#include <sstream>

template <typename T>

T get\_value(std::string&& message, bool(\*compare)(const T&) = compare\_default) {

T result;

std::string buff = "";

std::stringstream ss;

do {

ss.clear();

buff = "";

std::cout << message;

std::getline(std::cin, buff);

ss.str(buff);

ss >> result;

} while (ss.fail() || !compare(result));

return result;

};

template <typename T>

bool compare\_default(const T& value = 0) {

return true;

}

int main()

{

double first = get\_value<double>("Enter first number >\_ ");

double second = get\_value<double>("Enter second number >\_ ");

if (first < second)

{

std::cout << "Max = " << second << std::endl;

std::cout << "Min = " << first << std::endl;

}

else if (second < first)

{

std::cout << "Max = " << first << std::endl;

std::cout << "Min = " << second << std::endl;

}

else

{

std::cout << "Numbers are equel\n";

}

}

Задание №2.

Окно выполнения программы изображено на рис. 3.

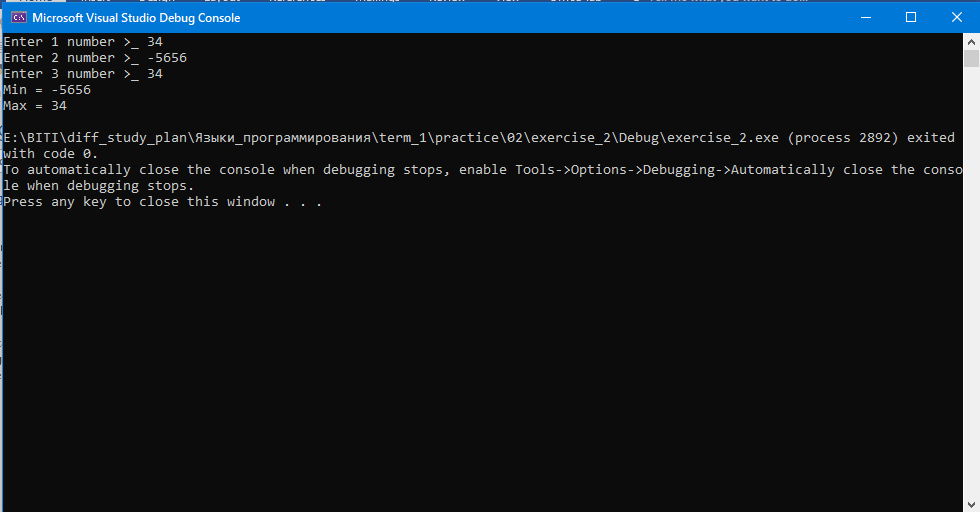


Рисунок 3 - Окно выполнение программы задания №2

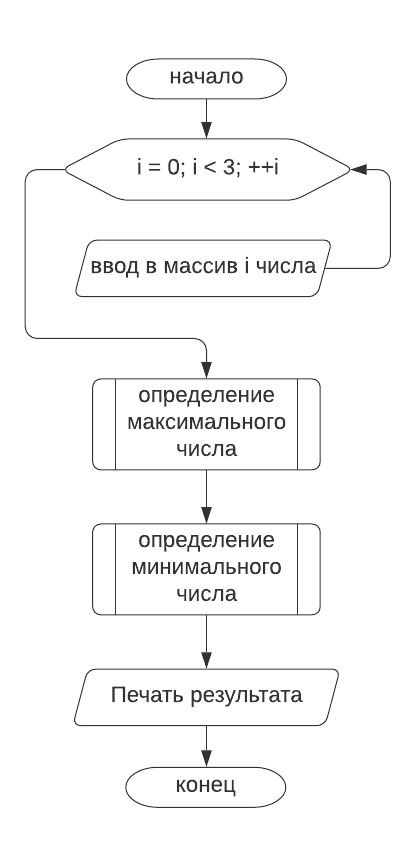
Блок схема алгоритма изображена на рис.4.

Рисунок 4 - Блок схема к заданию № 2

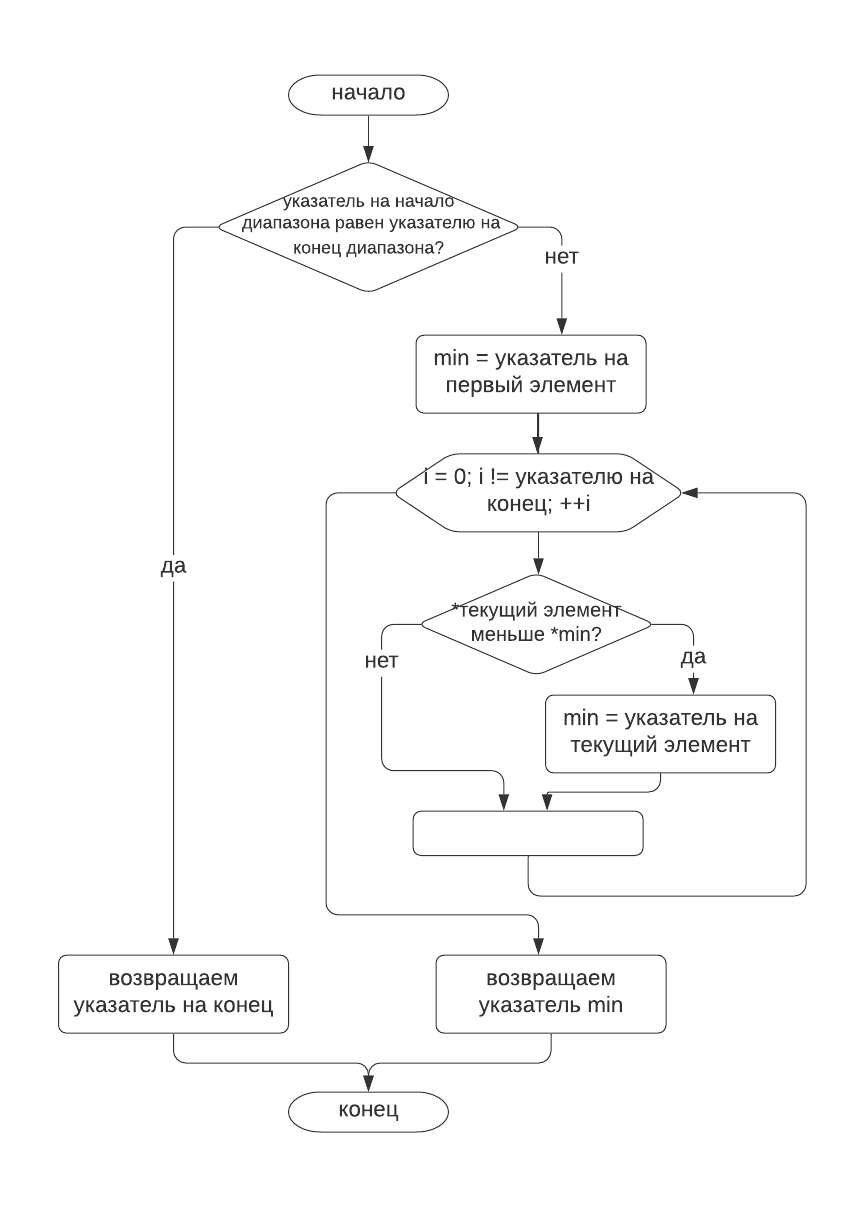
Блок схема функции, вычисляющей минимальный элемент представлена на рис. 5. Функция, вычисляющая максимальный элемент работает аналогично.

Рисунок 5 - Блок схема функции, вычисляющей минимальное значение

// created by a.nikolsky 02.12.2020

#include <iostream>

#include <sstream>

#include <vector>

template <typename T>

T get\_value(std::string&& message, bool(\*compare)(const T&) = compare\_default) {

T result;

std::string buff = "";

std::stringstream ss;

do {

ss.clear();

buff = "";

std::cout << message;

std::getline(std::cin, buff);

ss.str(buff);

ss >> result;

} while (ss.fail() || !compare(result));

return result;

};

template <typename T>

bool compare\_default(const T& value = 0) {

return true;

}

template<typename ForwardIt>

ForwardIt min\_item(ForwardIt first, ForwardIt last) {

if(first == last) return last;

ForwardIt smallest = first;

first++;

for (; first != last; ++first) {

if (\*first < \*smallest) {

smallest = first;

}

}

return smallest;

}

template<typename ForwardIt>

ForwardIt max\_item(ForwardIt first, ForwardIt last) {

if(first == last) return last;

ForwardIt biggest = first;

first++;

for (; first != last; ++first) {

if (\*biggest < \*first) {

biggest = first;

}

}

return biggest;

}

int main()

{

std::vector<double> v;

for (int i = 0; i < 3; ++i) {

v.push\_back(get\_value<double>(("Enter " +std::to\_string(i + 1) + " number >\_ ")));

}

std::cout << "Min = "<< \*min\_item(v.begin(), v.end()) << std::endl;

std::cout << "Max = "<< \*max\_item(v.begin(), v.end()) << std::endl;

}

Задание №3.

Окно выполнения программы изображено на рис.6.

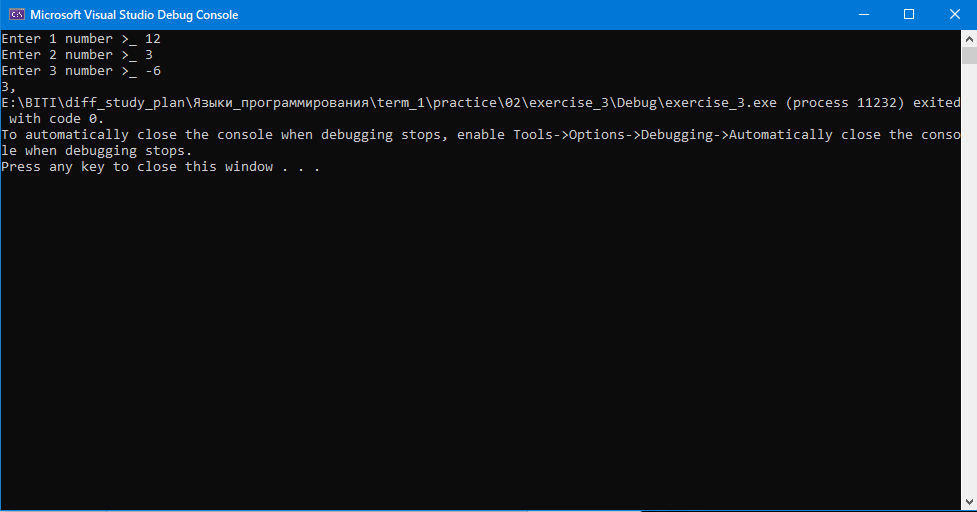


Рисунок 6 - Окно выполнение программы задания №3

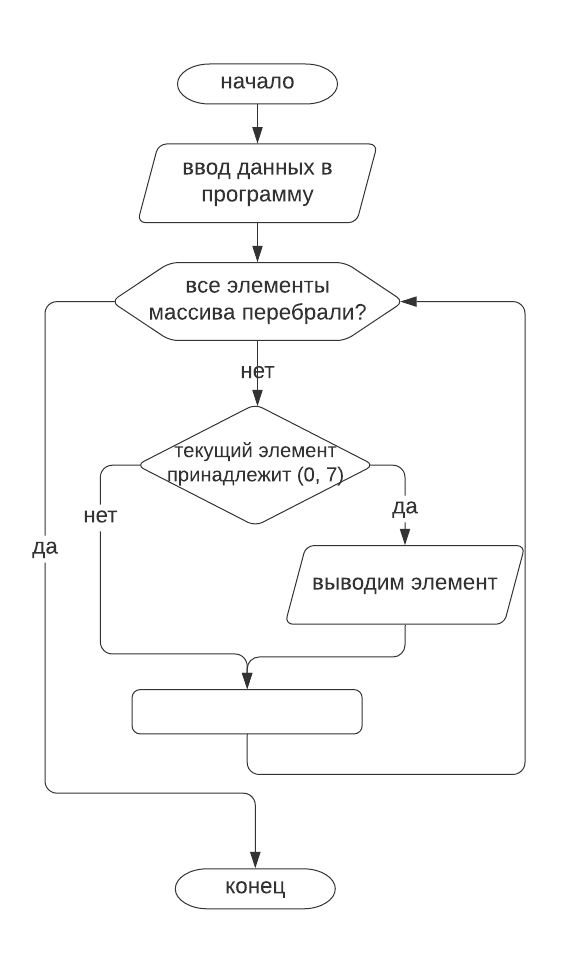
Блок схема алгоритма изображена на рис.7

Рисунок 7 - Блок схема к заданию №3

// created by a.nikolsky 02.12.2020

#include <iostream>

#include <sstream>

#include <vector>

template <typename T>

T get\_value(std::string&& message, bool(\*compare)(const T&) = compare\_default) {

T result;

std::string buff = "";

std::stringstream ss;

do {

ss.clear();

buff = "";

std::cout << message;

std::getline(std::cin, buff);

ss.str(buff);

ss >> result;

} while (ss.fail() || !compare(result));

return result;

};

template <typename T>

bool compare\_default(const T& value = 0) {

return true;

}

int main()

{

std::vector<double> v;

for (int i = 0; i < 3; ++i) {

v.push\_back(get\_value<double>(("Enter " + std::to\_string(i + 1) + " number >\_ ")));

}

for (const auto& it : v) {

if ( it > 1 && it < 7) {

std::cout << it << ", ";

}

}

}

Задание №4.

Окно выполнения программы изображено на рис. 8.

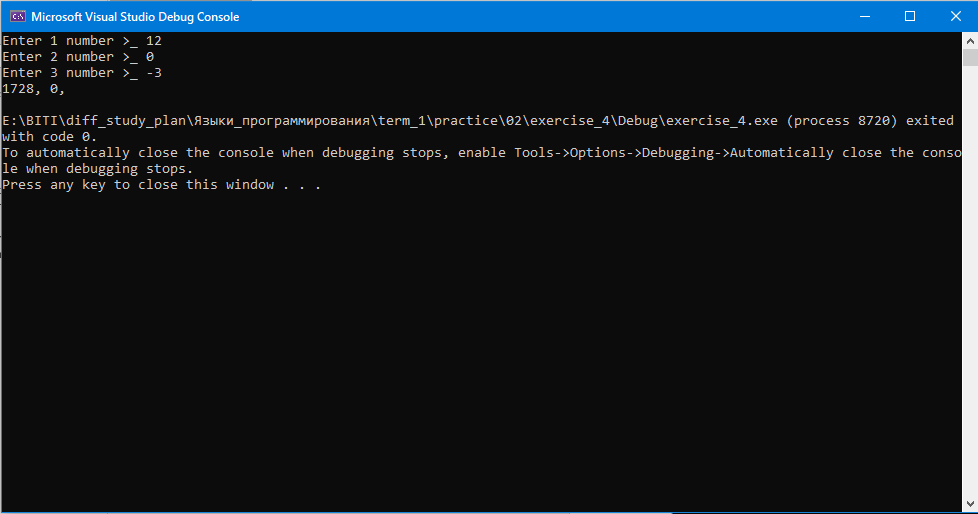


Рисунок 8 - Окно выполнение программы задания №4

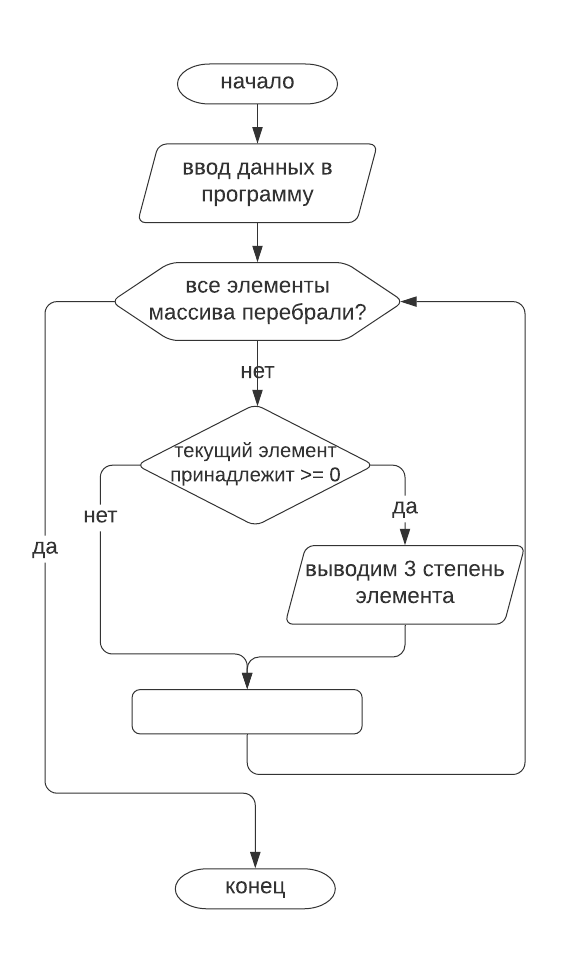
Блок схема алгоритма изображена на рис.9.

Рисунок 9 - Блок схема к заданию №4

// created by a.nikolsky 02.12.2020

#include <iostream>

#include <sstream>

#include <vector>

template <typename T>

T get\_value(std::string&& message, bool(\*compare)(const T&) = compare\_default) {

T result;

std::string buff = "";

std::stringstream ss;

do {

ss.clear();

buff = "";

std::cout << message;

std::getline(std::cin, buff);

ss.str(buff);

ss >> result;

} while (ss.fail() || !compare(result));

return result;

};

template <typename T>

bool compare\_default(const T& value = 0) {

return true;

}

int main()

{

std::vector<double> v;

for (int i = 0; i < 3; ++i) {

v.push\_back(get\_value<double>(("Enter " + std::to\_string(i + 1) + " number >\_ ")));

}

for (const auto& it : v) {

if (it >= 0 ) {

std::cout << pow(it, 3) << ", ";

}

}

std::cout << std::endl;

}

Задание №5

Окно выполнения программы изображено на рис. 10.

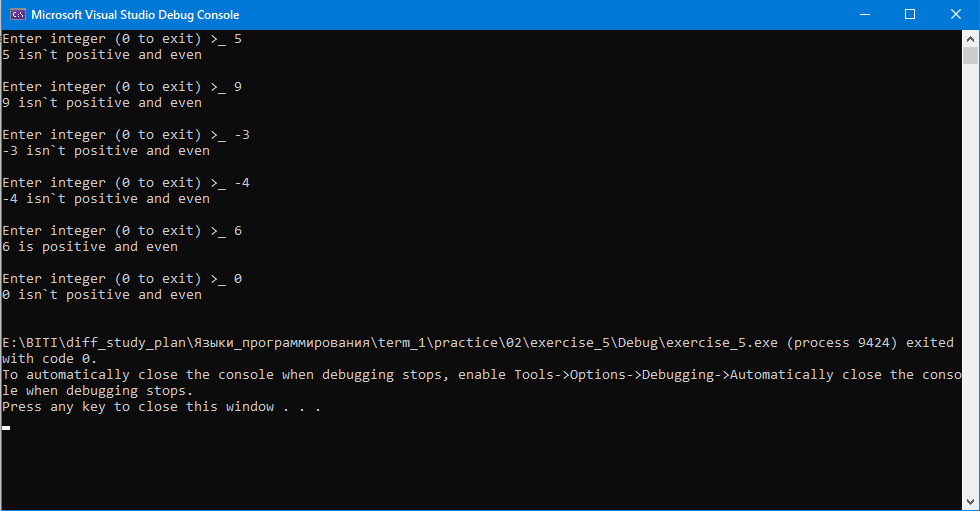


Рисунок 10 - Окно выполнение программы задания №5

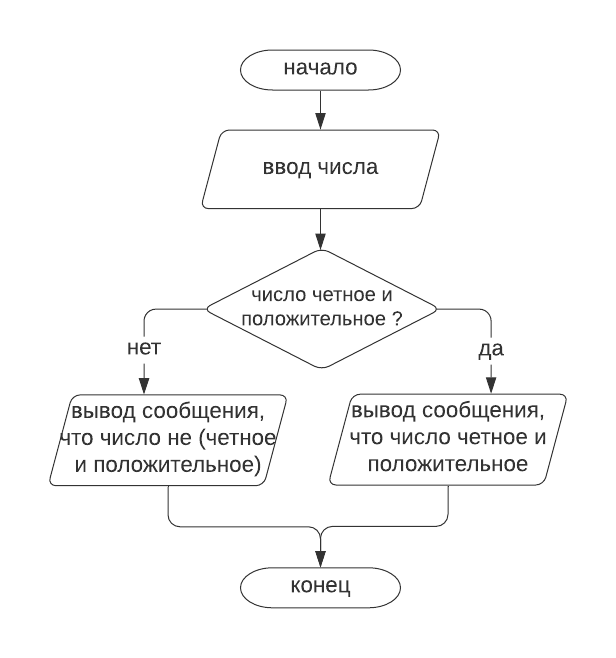
Блок схема алгоритма изображена на рис.11.

Рисунок 11 - Блок схема к заданию №5

// created by a.nikolsky 02.12.2020

#include <iostream>

#include <sstream>

#include <vector>

template <typename T>

T get\_value(std::string&& message, bool(\*compare)(const T&) = compare\_default) {

T result;

std::string buff = "";

std::stringstream ss;

do {

ss.clear();

buff = "";

std::cout << message;

std::getline(std::cin, buff);

ss.str(buff);

ss >> result;

} while (ss.fail() || !compare(result));

return result;

};

template <typename T>

bool compare\_default(const T& value = 0) {

return true;

}

bool is\_even(const int& number) {

return (number % 2 == 0 ? true : false);

}

bool is\_positive(const int& number) {

return (number > 0 ? true : false);

}

int main()

{

int number;

do {

number = get\_value<int>("Enter integer (0 to exit) >\_ ");

if (is\_positive(number) && is\_even(number)) {

std::cout << number << " is positive and even" << std::endl;

}

else {

std::cout << number << " isn`t positive and even" << std::endl;

}

std::cout << std::endl;

} while (number != 0);

}

Задание №6.

Окно выполнения программы изображено на рис. 12.

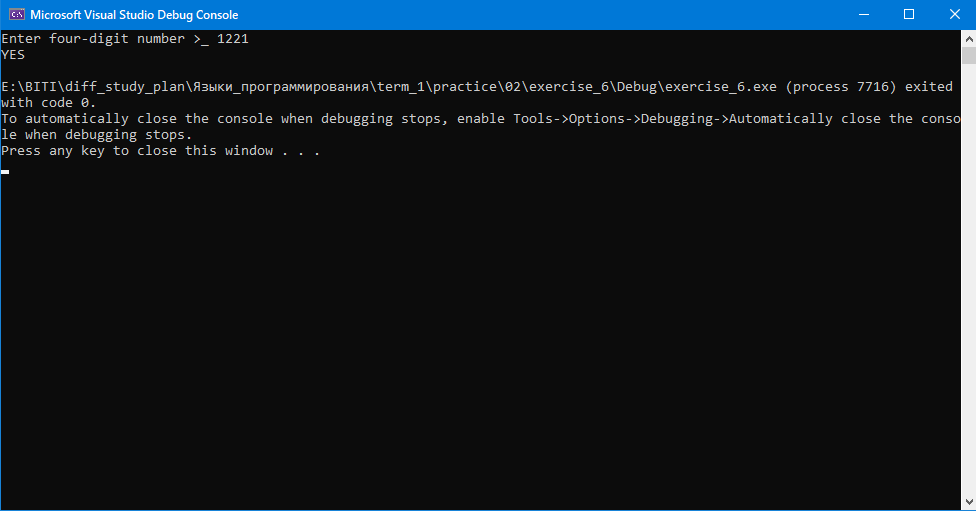


Рисунок 12 - Окно выполнение программы задания № 6

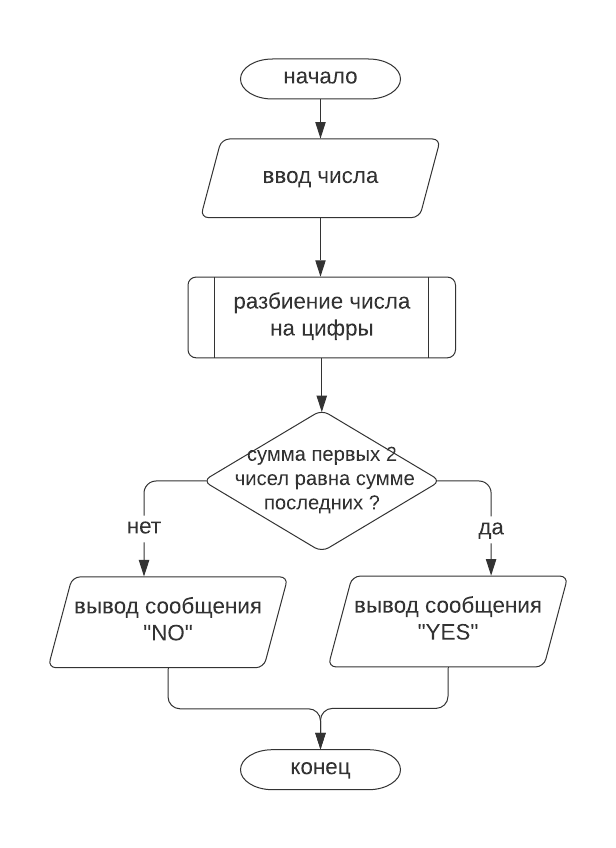
Блок схема алгоритма изображена на рис.13.

Рисунок 13 - Блок схема к заданию №6

// created by a.nikolsky 02.12.2020

#include <iostream>

#include <sstream>

#include <vector>

const int DIGIT\_COUNT = 4;

template <typename T>

T get\_value(std::string&& message, bool(\*compare)(const T&) = compare\_default) {

T result;

std::string buff = "";

std::stringstream ss;

do {

ss.clear();

buff = "";

std::cout << message;

std::getline(std::cin, buff);

ss.str(buff);

ss >> result;

} while (ss.fail() || !compare(result));

return result;

};

template <typename T>

bool compare\_default(const T& value = 0) {

return true;

}

bool compare\_digit\_count(const int& value) {

if (value >= pow(10, (DIGIT\_COUNT - 1)) &&

value <= pow(10, DIGIT\_COUNT) - 1 ) {

return true;

}

else return false;

}

template <typename FIterator>

long int sum(FIterator first, FIterator last) {

long sum = 0;

for (; first != last; ++first) {

sum += \*first;

}

return sum;

}

int main() {

int number = get\_value<int>("Enter four-digit number >\_ ", compare\_digit\_count);

std::vector<int> v;

for (int i = pow(10, (DIGIT\_COUNT - 1)); i > 0; i /= 10) {

v.push\_back(number / i);

number = number % i;

}

if (sum(v.begin(), v.begin() + v.size() / 2) ==

(sum(v.rbegin(), v.rbegin() + v.size() / 2))) {

std::cout << "YES" << std::endl;

}

else

std::cout << "NO" << std::endl;

}

Задание №7.

Окно выполнения программы изображено на рис. 13.

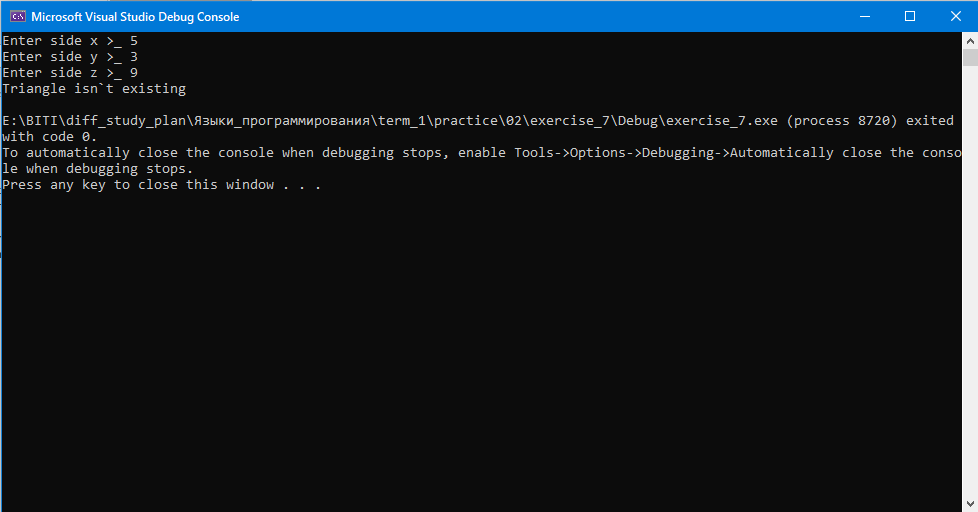


Рисунок 13 - Окно выполнение программы задания № 7

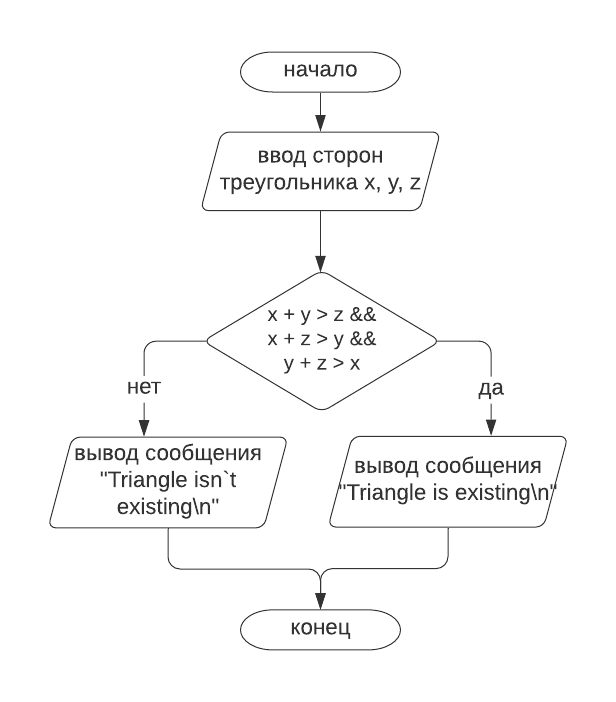
Блок схема алгоритма изображена на рис.14.

Рисунок 14 - Блок схема к заданию №7

// created by a.nikolsky 02.12.2020

#include <iostream>

#include <sstream>

#include <vector>

template <typename T>

T get\_value(std::string&& message, bool(\*compare)(const T&) = compare\_default) {

T result;

std::string buff = "";

std::stringstream ss;

do {

ss.clear();

buff = "";

std::cout << message;

std::getline(std::cin, buff);

ss.str(buff);

ss >> result;

} while (ss.fail() || !compare(result));

return result;

};

template <typename T>

bool compare\_default(const T& value = 0) {

return true;

}

bool side\_compare(const double& side) {

return (side > 0 ? true : false);

}

int main (){

double x = get\_value<double>("Enter side x >\_ ", side\_compare);

double y = get\_value<double>("Enter side y >\_ ", side\_compare);

double z = get\_value<double>("Enter side z >\_ ", side\_compare);

if (x + y > z && x + z > y && y + z > x) {

std::cout << "Triangle is existing\n";

}

else {

std::cout << "Triangle isn`t existing\n";

}

}

*Задание 9 для 9 варианта практически аналогично заданию 1. Код написал. Блок-схему и прочее не рисовал.*

Вывод: в данной практической работе изучили алгоритмы ветвления, научились применять условный оператор и оператор варианта в языке программирования СИ.