**Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**"Уфимский государственный авиационный технический университет"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Программирование

**Отчет по лабораторной работе №3**

**Тема: «** Указатели. Передача параметров в функцию по указателю. Сортировка одномерного массива. Структуры »

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМИ-148 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Фаизова А. Н. |  |  |  |
| Принял | Гайнетдинова А.А. |  |  |  |

**Уфа 2017**

**Требования к составу отчета:**

Отчет к лабораторной работе оформляется в текстовом процессоре Microsoft Word или OpenOffice (LibreOffice) Write в соответствии с требованиями стандарта СТО УГАТУ 016-2007 и содержать

• титульный лист,

• описание цели работы и краткую теоретическую справку по использованным операторам и функциям языка программирования,

• описание выполнения задания:

1. формулировка задания, как в методичке,
2. блок-схема каждой разработанной функции,
3. исходный текст разработанного приложения,
4. скриншоты примеров выполнения программы.

• выводы

• список использованной литературы,

**Требования к оформлению отчета**

* Текст отчета набирается шрифтом Times New Roman, размер шрифта 14pt или 12pt единый во всем документе. Исходный код должен быть набран любым моноширинным шрифтом (например, Courier New), размер символов можно уменьшить до 10 pt. Поля страницы: верхнее и нижнее – 2 см., левое – 2 см, правое – 1.5 см.
* Абзац должен начинаться с красной строки, за исключением тех случаев, когда абзац разорван каким-либо математическим выражением. Выравнивание внутри текстовых абзацев «по ширине», выравнивание в блоке с исходным кодом – «по левому краю».
* Все блок-схемы оформляются в соответствие с ГОСТ 19.701. Описание ГОСТ и пример блок-схемы приведены в разделе Справочник, методических указаний по выполнению лабораторных работ на сайте. Если блок-схемы составляются в сторонних приложениях, следует предусмотреть возможность их правки в учебном классе. В противном случае отчет с некорректно составленной блок-схемой и исправлениями вручную приниматься не будут.
* Описание выполнения каждого задания начинается с новой страницы. Следует использовать заголовок с текстом «**Индивидуальное задание №\_\_**», набранный полужирным шрифтом с выравниванием по левому краю, без красной строки. Текст задания, текст в блок-схеме и описание программы следует набирать шрифтом Times New Roman с прямым начертанием (не курсив!!!).
* Текст на скриншотах должен быть читаемым. Скриншоты должны содержать только окно с результатами выполнения программы, а не весь рабочий стол. Размер шрифта на скриншотах должен соответствовать по размеру окружающему его тексту.

Ниже и выше красным выделены места, которые необходимо изменить.

**Цель:** изучить принципы работы с указателями и структурными типами данных (массивы и структуры) в СИ. Приобрести практические навыки в реализации алгоритмов сортировки одномерных массивов.

**Теоретический материал**

*Указатель* ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *pointer*)  это переменная, хранящая адрес некой области памяти. Часто говорят, что указатель указывает на область памяти, или указатель ссылается на область памяти.

В области памяти, на которую ссылается указатель может располагаться некая переменная, массив данных или просто какие-либо бинарные данные, например изображение. Поскольку нет никакого стандартного типа данных, позволяющего хранить в переменной этого типа графическое изображение, звук или видео, то подобную информацию можно загрузить в память и хранить объем занятой памяти, а также адрес первой ячейки памяти всей области. Вот для этого и используются указатели, поскольку они позволяют хранить адрес в памяти.

Если указатель ссылается на область памяти, в которой хранится переменная стандартного типа (int, float, char и т.д.), то такой указатель называется типизированным. Типизированный указатель описывается также, как и обычные переменные, но перед именем переменной-указателя ставится значок "звездочка" (\*)

*Массив* - это пронумерованная совокупность однотипных данных. Каждое значение в массиве называется элементом массива, а номер элемента массива - индексом.

Если каждое значение в совокупности имеет только один индекс (номер), необходимый, для того, чтобы обратиться к нему, то массив называется одномерным. Типичным примером одномерного массива является вектор. Если для обращения к элементу массива необходимо указать два индекса, то массив называется двумерным. Типичным примером двумерного массива является матрица.

Кроме числовых и символьных данных в C++ есть тип, который позволяет объединить разнородные данные и обрабатывать их как единое целое. Этот тип называется *структурой*. Структура является собранием одного или более объектов (переменных, массивов, указателей и т.д.), которые для удобства работы с ними объединены под одним именем. Переменные, входящие в состав структуры, называются полями

Схема описания структуры выглядит следующим образом:

struct <имя структуры>[: наследования] {

<тип поля> <имя\_поля1> :[кол-во бит];

<тип поля> <имя\_поля2> :[кол-во бит];

...

<тип поля> <имя\_поляN> :[кол-во бит];

} <глобальные переменные>;

*Сортировка* — это алгоритм для упорядочивания элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число. Существует три вида сортировки: метод пузырька, простого выбора и метод вставок.

Mетод вставок

Линейная вставка

Элементы массива делятся на уже упорядоченную последовательность а0, а1, …, аi-1 и неупорядоченную аi, ai+1, …,an-1. В каждом проходе из неупорядоченной последовательности извлекается элемент аi (в первом проходе i=1) и вставляется в упорядоченную последовательность из i элементов без нарушения упорядоченности в ней. Этот алгоритм повторяется для i=2,3,…,n-1. Алгоритм вставки аi в упорядоченную последовательность из i элементов заключается в продвижении вставляемого элемента в начало последовательности, сравнивая его с аi-1, ai-2 и т. д. Продвижение заканчивается на элементе аj<=ai или при прохождении всей последовательности.

Бинарная вставка.

В этом методе, в отличие от линейной вставки, для отыскания места для вставки элемента ai в упорядоченную последовательность используется алгоритм бинарного поиска, при котором элемент ai сравнивается со средним элементом упорядоченной последовательности, а затем процесс деления пополам продолжается до тех пор, пока не будет найдена точка включения.

rand – функция, которая записывает в массив комбинацию случайных чисел из задаваемого пользователем диапазона.

**Ответы на контрольные вопросы**

1. Дайте определение указателя.

Указатель это переменная, хранящая адрес некой области памяти. Часто говорят, что указатель указывает на область памяти, или указатель ссылается на область памяти.

1. Каким образом можно получить адрес переменной в явном виде?

Чтобы получить адрес переменной в явном виде используют унарный оператор &. Оператор неприменим к выражениям, константам, битовым полям, регистровым переменным или внешним объектам (файлам).

1. Опишите передачу параметров в функцию по указателю.

Передача параметра функции по указателю нужна, когда в качестве аргумента функции выступает массив или вы не хотите использовать ссылку.

Указатели на функции как параметры позволяют создавать функции, реализующие тот или иной метод обработки другой функции, которая заранее не определена.

Если передавать параметры через указатель, код функции выглядит так:

double function(double \* setting1, double \* setting2);

1. Дайте определение массива.

Массив - это пронумерованная совокупность однотипных данных. Каждое значение в массиве называется элементом массива, а номер элемента массива - индексом.

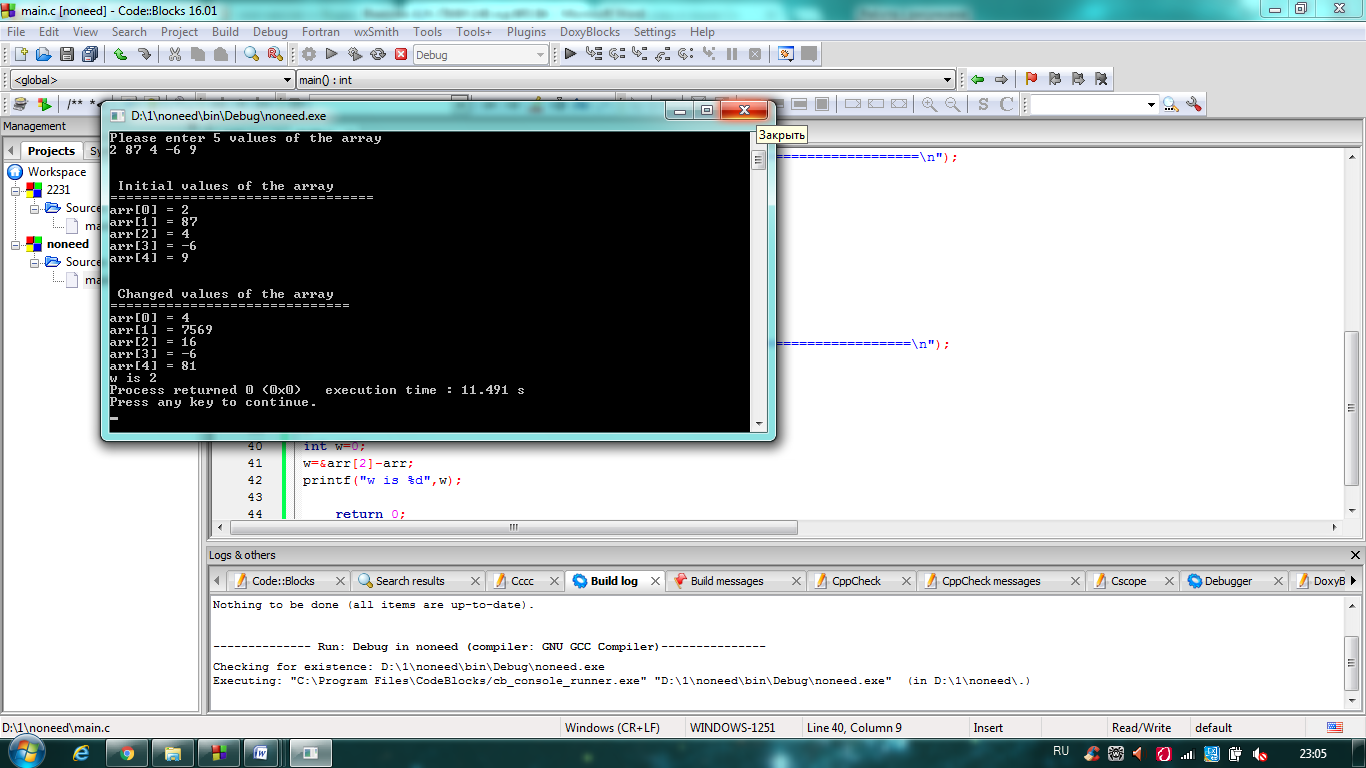
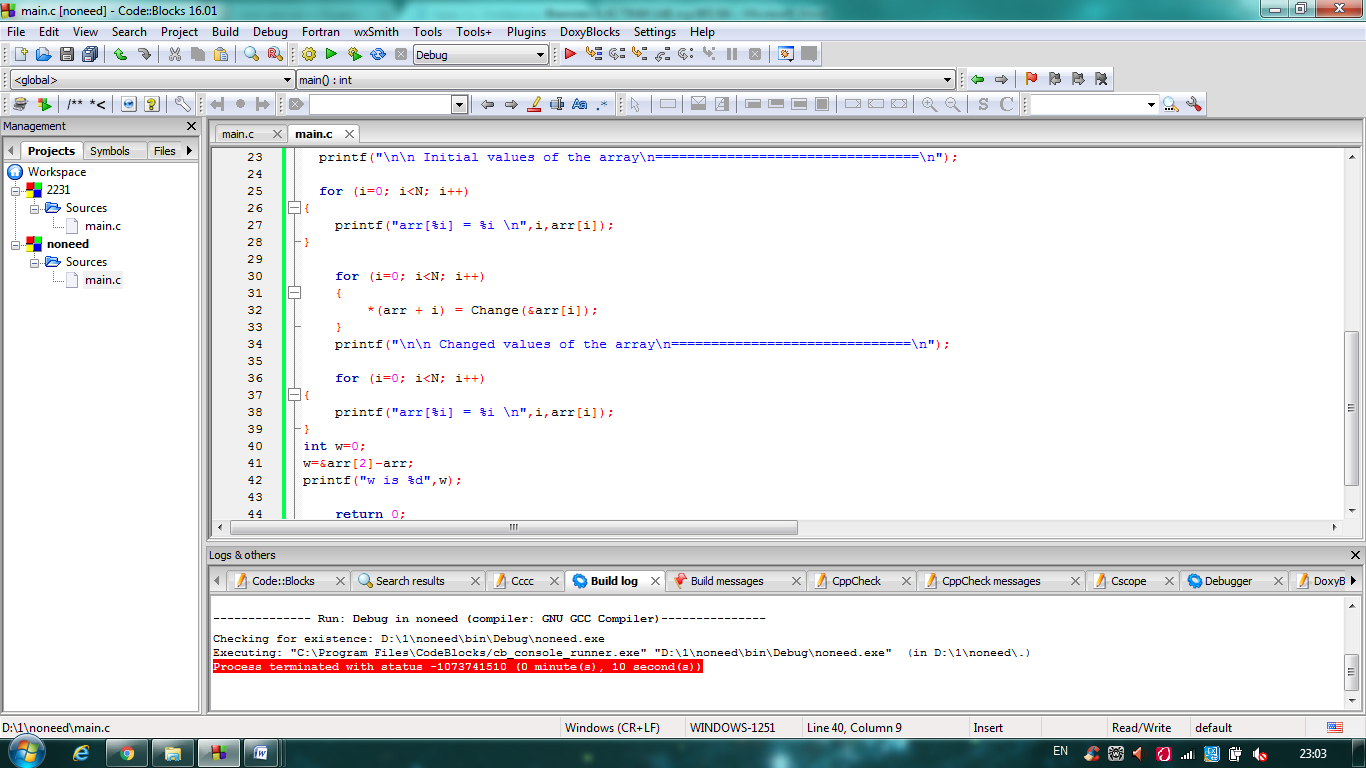
1. Какой результат будет получен при разыменовании имени массива?

Имя массива без индексов в Си является адресом его первого элемента. Соответственно через операцию разыменования мы можем получить значение по этому адресу.

1. Что происходит при добавлении целочисленного значения n к указателю, адресующему некоторый элемент в массиве?

Пусть ptr – указатель, ссылающийся на элементы массива Arr. Тогда выражение \*(ptr+n) будет ссылаться на значение, содержащееся в элементе Arr[n].

1. Каков результат вычитания из адреса любого элемента массива имени массива?



Разница адреса любого элемента и адреса первого элемента.

Имя массива всегда указывает только на первый элемент массива.

1. В чем отличие статического и динамического выделения памяти для массива?

Если размер массива задается при его описании, то такие массивы называются статическими. Размер таких массивов нельзя изменить в процессе работы программы.

Однако разработчик не всегда знает каков будет размер поступающих в программу данных. Например, никогда нельзя точно сказать сколько человек будет в группе на момент начала занятий. Для первого курса это означает, что может поступить больше или меньше человек, чем планировалось. Для старших курсов это означает, что после сессии кого-то могут отчислить или восстановить.

Массивы, размер которых не задается при их описании и может быть установлен и/или изменен в ходе работы программы, называются динамическими.

1. В чем отличие функций динамического выделения памяти calloc() и malloc()?

В отличии от malloc() функция calloc() кроме выделения области памяти под массив объектов еще производит инициализацию элементов массива нулевыми значениями.

Функция calloc()

#include <stdlib.h>

void \*calloc(size\_t num, size\_t size);

Функция calloc() выделяет память, размер которой равен значению выражения num \* size, т.е. память, достаточную для размещения массива, содержащего num объектов размером size . Выделенная область памяти обнуляется. Функция calloc() возвращает указатель на первый байт выделенной области памяти для массива num объектов, каждый из которых имеет размер size или NULL, если запрос на память выполнить нельзя. Если для удовлетворения запроса нет достаточного объема памяти, возвращается нулевой указатель. Перед попыткой использовать распределенную память важно проверить, что возвращаемое значение не равно нулю. Тип void может быть переопределен для требуемого типа, т.е. для char, int, float, double.

Функция malloc()

#include <stdlib.h>

void \*malloc(size\_t size);

Функция malloc() возвращает указатель на первый байт области памяти размера size, которая была выделена из динамически распределяемой области памяти. Если для удовлетворения запроса в динамически распределяемой области памяти нет достаточного объема памяти, возвращается нулевой указатель NULL. При этом следует иметь в виду, что попытка использовать нулевой указатель обычно приводит к полному отказу системы. Выделенная область памяти не инициализируется.

**Индивидуальное задание №1**

Задание: *Написать программу, вычисляющую значения ab и ba. Расчет должен производиться в функции, которая получает данные по указателю и возвращает результат по указателю.*

Блок-схема

Начало

Ввод

a

Ввод

b

x:=1

y:=1

m:=|b|

**WL1**

Пока m>0

x:=x\*a

**A**

Начало

алгоритма

Ввод первого числа

Ввод второго числа

Переменным, отвечающим за значения получаемых чисел, присваиваем начальные значения

Вспомогательную переменную m приравниваем к модулю значения второго числа

Начало цикла, включающего в себя функцию по вычислению значения

ab

**A**

m:=m-1

**WL1**

b < 0

x:=

Да

Нет

Вывод

x

Соединитель

Конец цикла

Проверка условия «Отрицательна ли степень?»

Присвоение x обратного значения

при отриц.степени

Ввод значения ab

m:=|а|

**WL2**

Пока m>0

x:=x\*b

**B**

Начало цикла, включающего в себя функцию по вычислению значения

ba

m:=m-1

**WL2**

**B**

b < 0

Да

Нет

Конец

Конец алгоритма

y:=

Вывод

y

Ввод значения ba

Присвоение y обратного значения

при отриц.степени

Проверка условия «Отрицательна ли степень?»

Конец цикла

Описание программы**:**

a – первое целочисленное число;

b - второе целочисленное число;

x – значение ab;

y – значение ba;

pa – указатель на первое число;

pb – указатель на второе число;

px – указатель на переменную, содержащую значение первое\_числовторое\_число;

m - вспомогательная переменная, с использование которой происходит выполнение цикла в функции по возведению первое\_число в степень второе\_число;

double Pow(int \*указатель1, int \*указатель2, double \*указатель3) – функция возводящая указатель1 в степень указатель2, полученное значение присваивается в указатель3 и возвращается в основную функцию;

Исходный код программы

#include <stdio.h>

#include <stdlib.h>

double Pow(int \*pa,int \*pb, double\* px)

{

int m = abs((\*pb));

while (m > 0)

{

\*px \*= (double)(\*pa);

m--;

}

if ((\*pb) < 0)

{\*px = 1/(\*px);}

return (\*px);

}

int main()

{

double x = 1,y = 1;

int a = 0, b = 0;

printf ("Enter the first integer number:\n");

scanf ("%d", &a);

printf ("Enter the second integer number:\n");

scanf ("%d", &b);

x = Pow(&a,&b,&x);

printf ("The first number in the power of the second number: %lf \n",x);

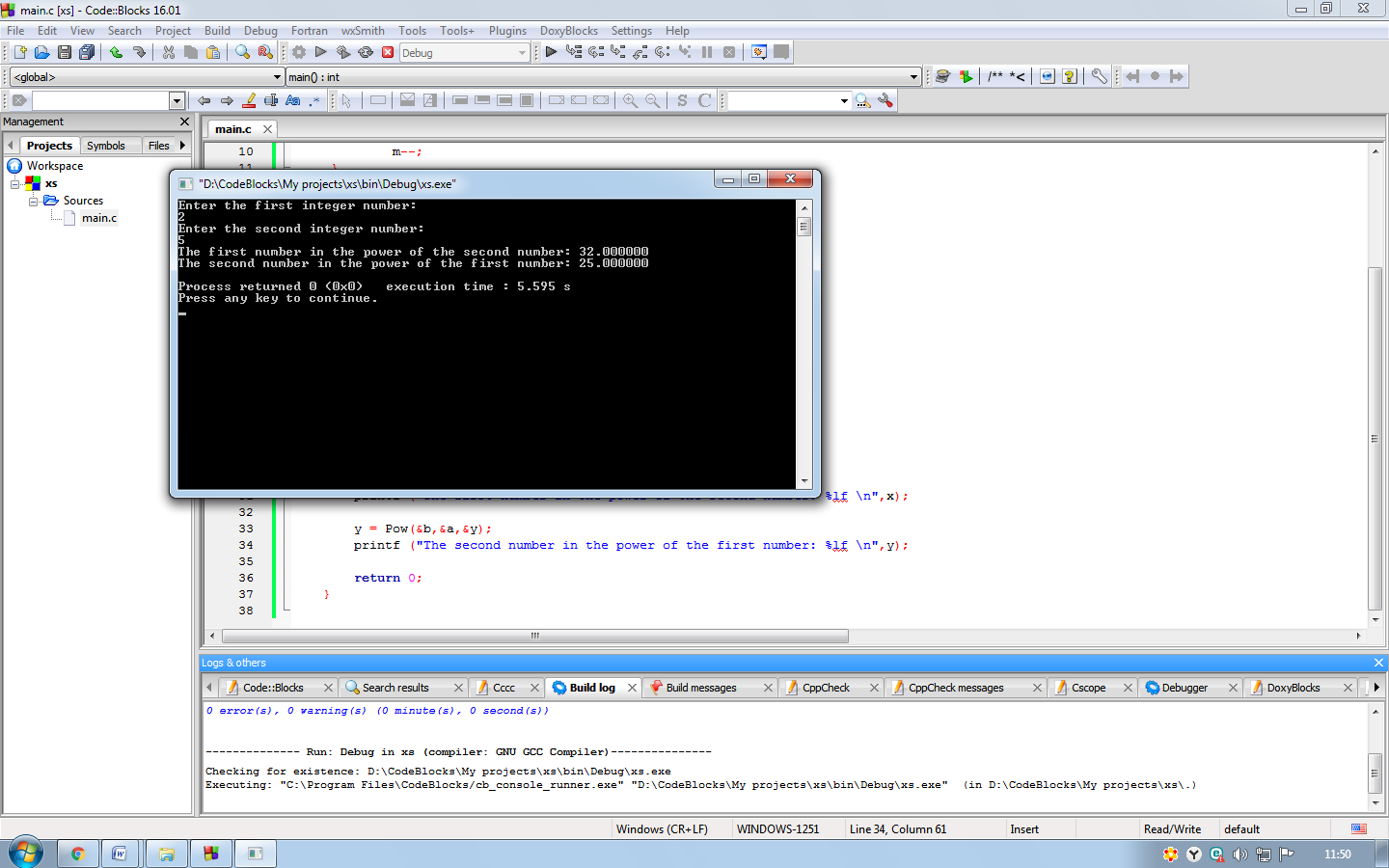
y = Pow(&b,&a,&y);

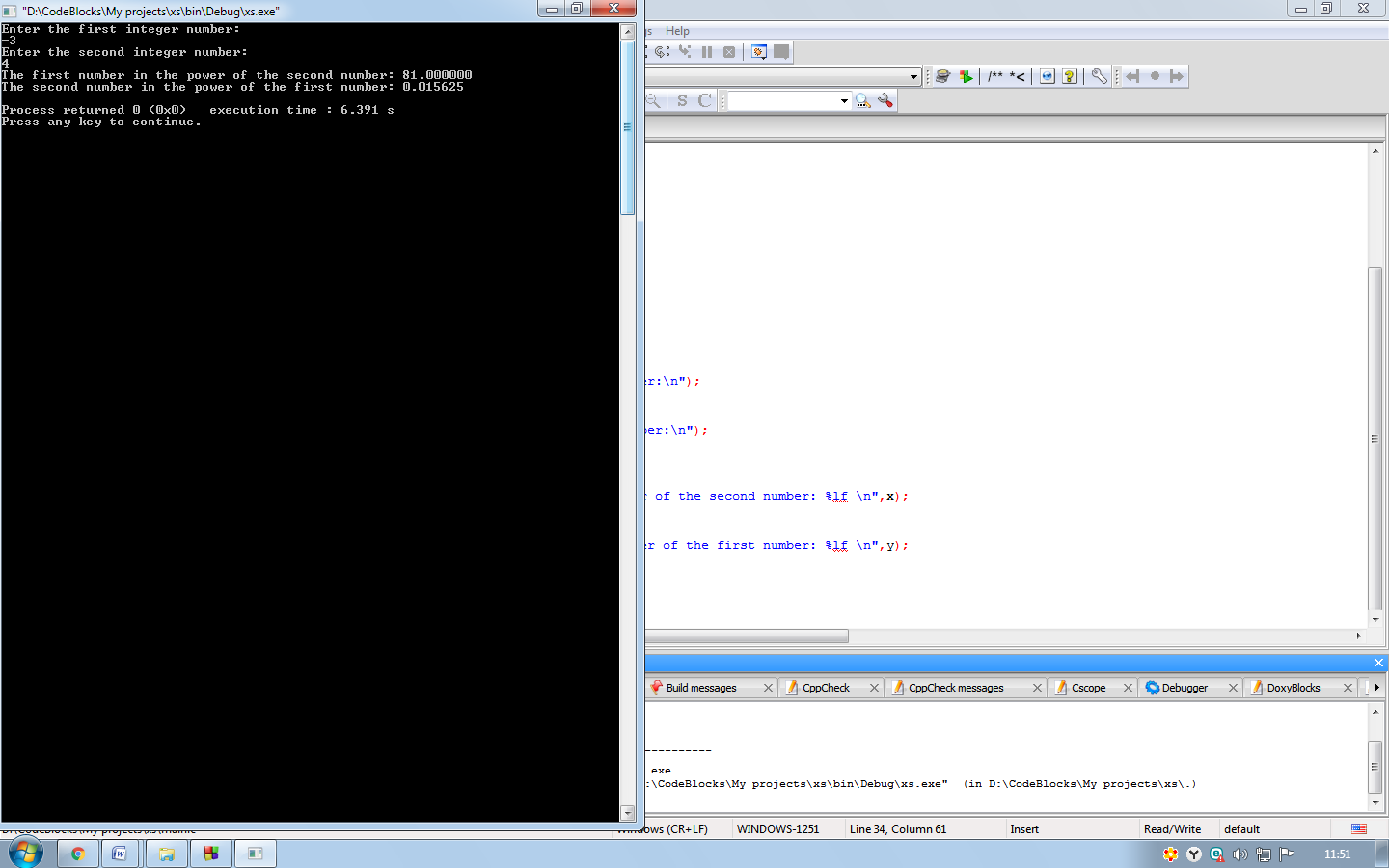
printf ("The second number in the power of the first number: %lf \n",y);

return 0;

}

Пример выполнения программы





**Индивидуальное задание №2**

Задание: *Написать программу, заменяющую положительные элементы массива на их квадраты. При вводе/выводе элементов использовать индексы, а обработку элементов массива осуществлять с помощью указателей. Расчет должен производиться в функции, в которую массив передается по указателю.*

Блок-схема

Начало

**А**

i:=1,N;1

Ввод

arri

Вывод

arri

i:=1,N;1

i:=1,N;1

arri:=Change(arri)

**А**

i:=1,N;1

Вывод

arri

Конец

Начало функции Change

arri > 0

arri:=arri\*arri

Конец функции Change

да

нет

Описание программы**:**

i –переменная, отвечающая за индекс элемента массива, и переменная цикла;

N –количество элементов массива;

arr[N] –массив элементов;

int Change(int \*arr) –функция, производящая замену положительных элементов массива на их квадраты ;

Исходный код программы

#include <stdio.h>

#define N 5

int Change(int \*arr)

{

if (\*arr > 0)

{

\*arr = (\*arr)\*(\*arr);

}

return (\*arr);

}

int main()

{

int i = 0;

int arr[N];

printf("Please enter %d values of the array \n",N);

for (i=0; i<N; i++)

{

scanf("%d", &arr[i]);

}

printf("\n\n Initial values of the array\n=================================\n");

for (i=0; i<N; i++)

{

printf("arr[%i] = %i \n",i,arr[i]);

}

for (i=0; i<N; i++)

{

\*(arr + i) = Change(&arr[i]);

}

printf("\n\n Changed values of the array\n==============================\n");

for (i=0; i<N; i++)

{

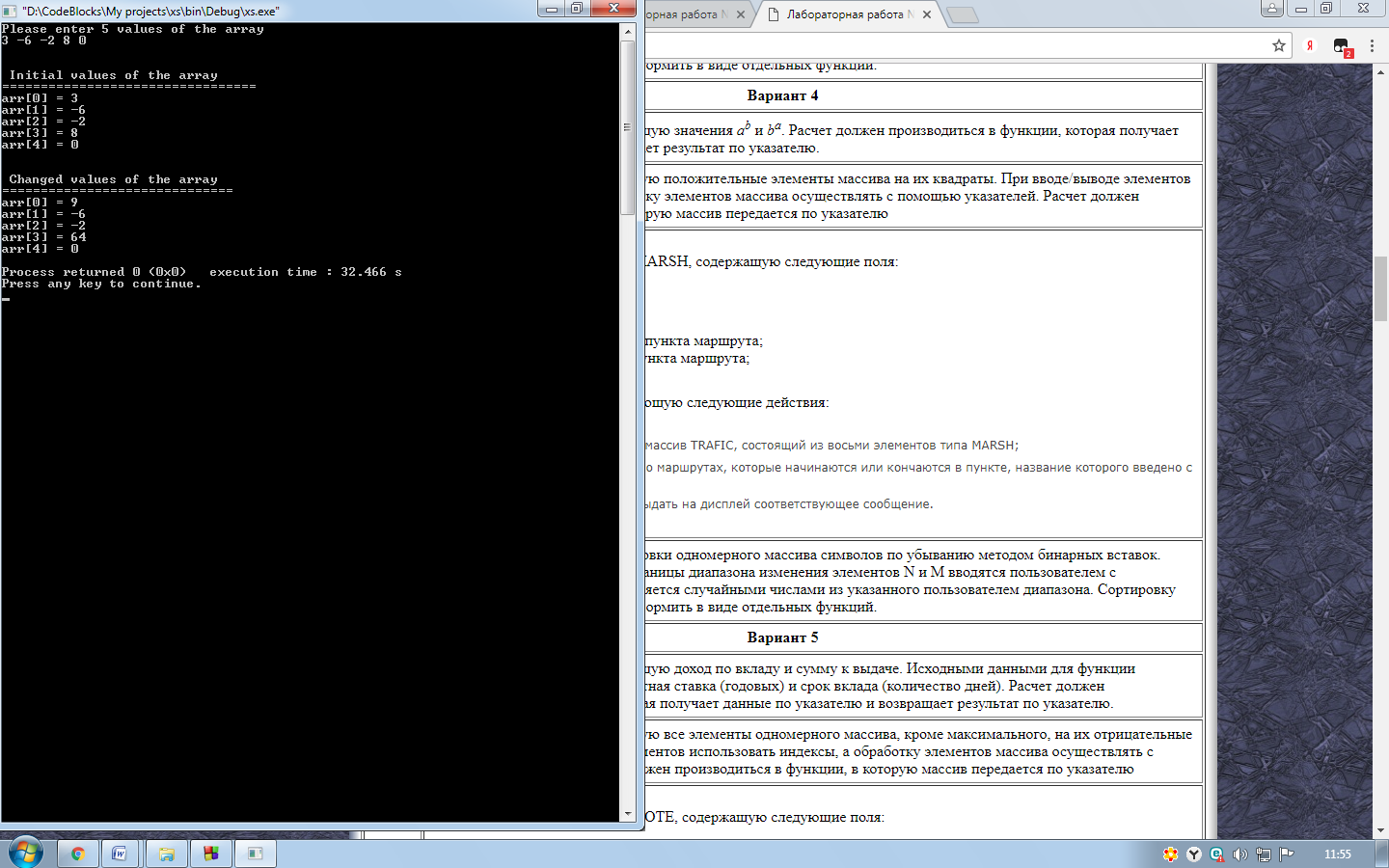
printf("arr[%i] = %i \n",i,arr[i]);

}

return 0;

}

Пример выполнения программы



**Индивидуальное задание №3**

Задание: *Описать структуру с именем MARSH, содержащую следующие поля:*

*NUMER – номер маршрута;*

*TIME – среднее время в пути;*

*BEGST – название начального пункта маршрута;*

*TERM – название конечного пункта маршрута;*

*Написать программу, выполняющую следующие действия:*

* *ввод с клавиатуры данных в массив TRAFIC, состоящий из восьми элементов типа MARSH;*
* *вывод на экран информации о маршрутах, которые начинаются или кончаются в пункте, название которого введено с клавиатуры;*
* *если таких маршрутов нет, выдать на дисплей соответствующее сообщение.*

Описание программы**:**

MARSH – имя структуры, содержащей следующие поля:

NUMER – номер маршрута;

TIME – среднее время в пути;

BEGST – название начального пункта маршрута;

TERM – название конечного пункта маршрута;

TRAFIC – массив, состоящий из восьми элементов типа MARSH

N – количество элементов типа MARSH (в нашем случае 8);

name – массив, в который записывается название пункта маршрута, введённое с клавиатуры (впоследствии необходимо вывести согласно этому названию информацию о маршрутах, начинающихся или заканчивающихся в введённом пункте);

i – переменная цикла;

Исходный код программы

#include <string.h>

#include <stdio.h>

#define N 8

typedef struct {

int NUMER;

int TIME;

char BEGST[20];

char TERM[20];

} MARSH;

MARSH TRAFIC[N] = {

{1, 46,"FirstStart", "FirstEnd"},

{2, 48,"SecondStart", "SecondEnd"},

{3, 39,"ThirdStart", "EighthStart"},

{4, 43,"SecondEnd", "ForthEnd"},

{5, 58,"FifthStart", "FifthEnd"},

{6, 49,"SixthStart", "FirstStart"},

{7, 56,"SeventhStart", "SeventhEnd"},

{8, 38,"EighthStart", "SecondEnd"}

};

int main(){

char name[20];

int i=0;

int result1,result2;

int count = 0;

printf("Please enter point of the route: ");

scanf("%s",name);

printf("\n \n====================================================");

for(i=0; i<N; i++){

result1 = strcmp( TRAFIC[i].BEGST, name);

result2 = strcmp( TRAFIC[i].TERM, name);

if (result1 == 0 || result2 == 0)

{

count++;

printf("\n Name of the route: %d;\n \n Time of the route: %d;\n \n Starting point: %s;\n \n Ending point: %s;\n \n====================================================", TRAFIC[i].NUMER,TRAFIC[i].TIME, TRAFIC[i].BEGST, TRAFIC[i].TERM);

};

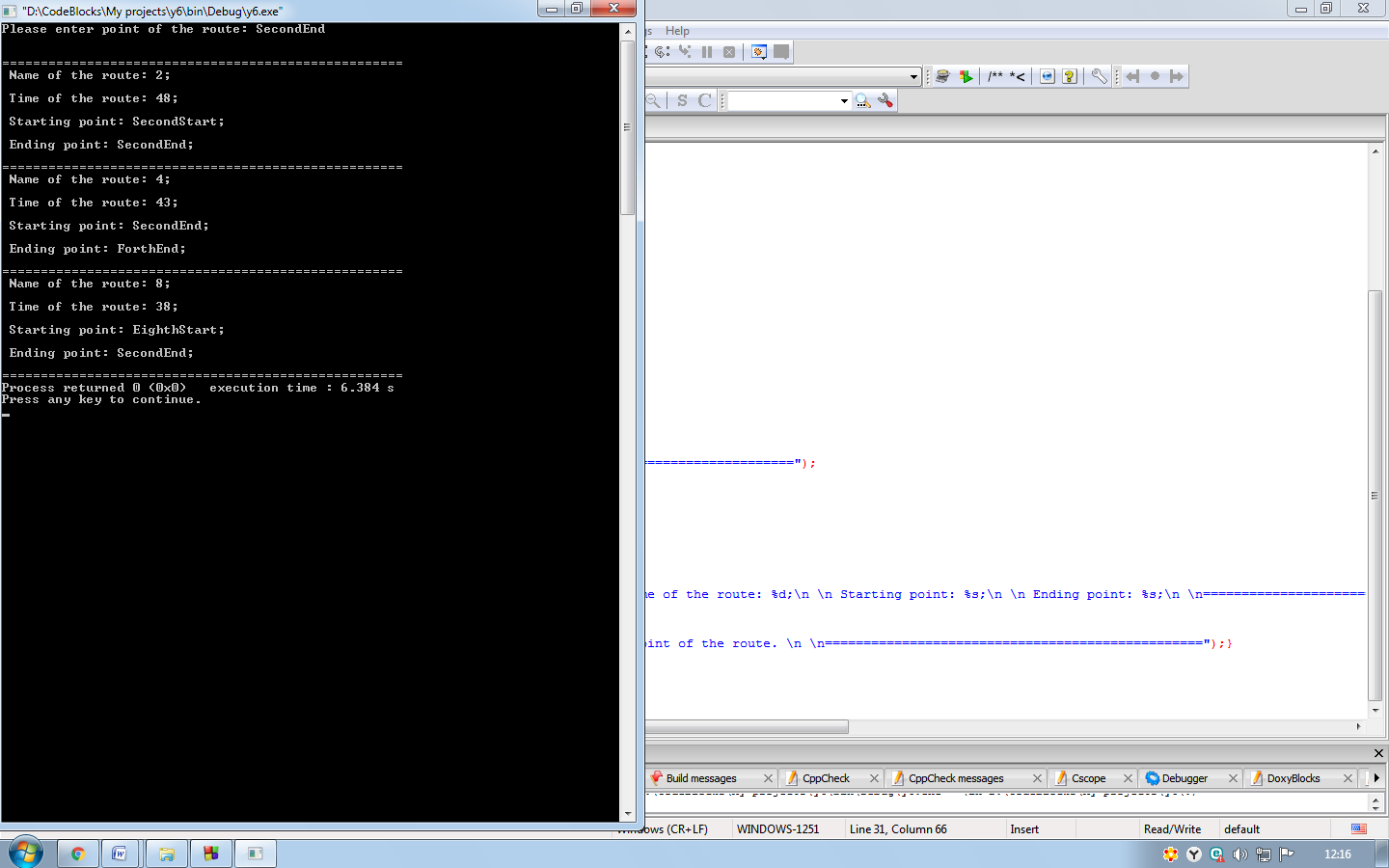
}

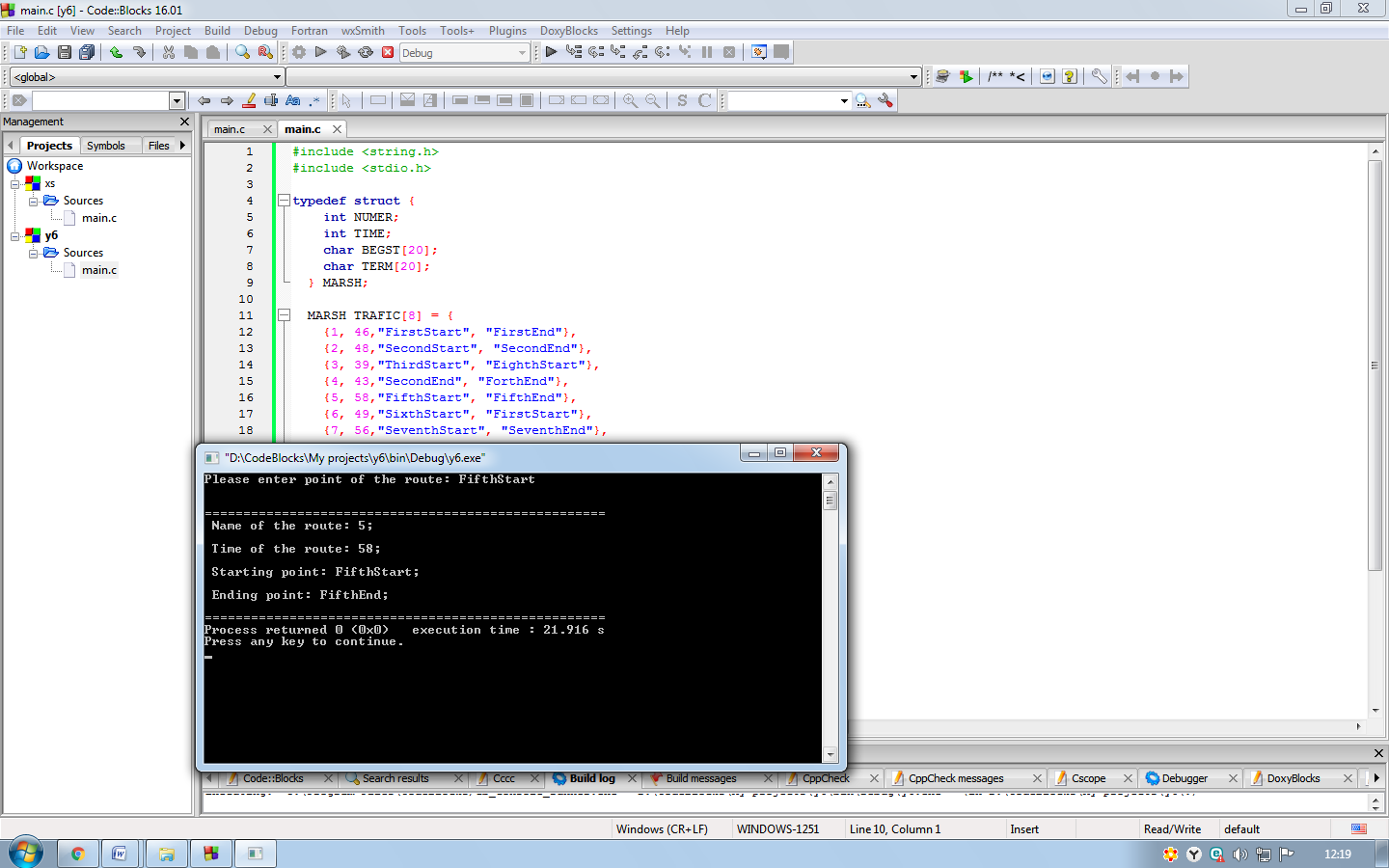
if (count == 0){printf("\n There is no such point of the route. \n \n=================================================");}

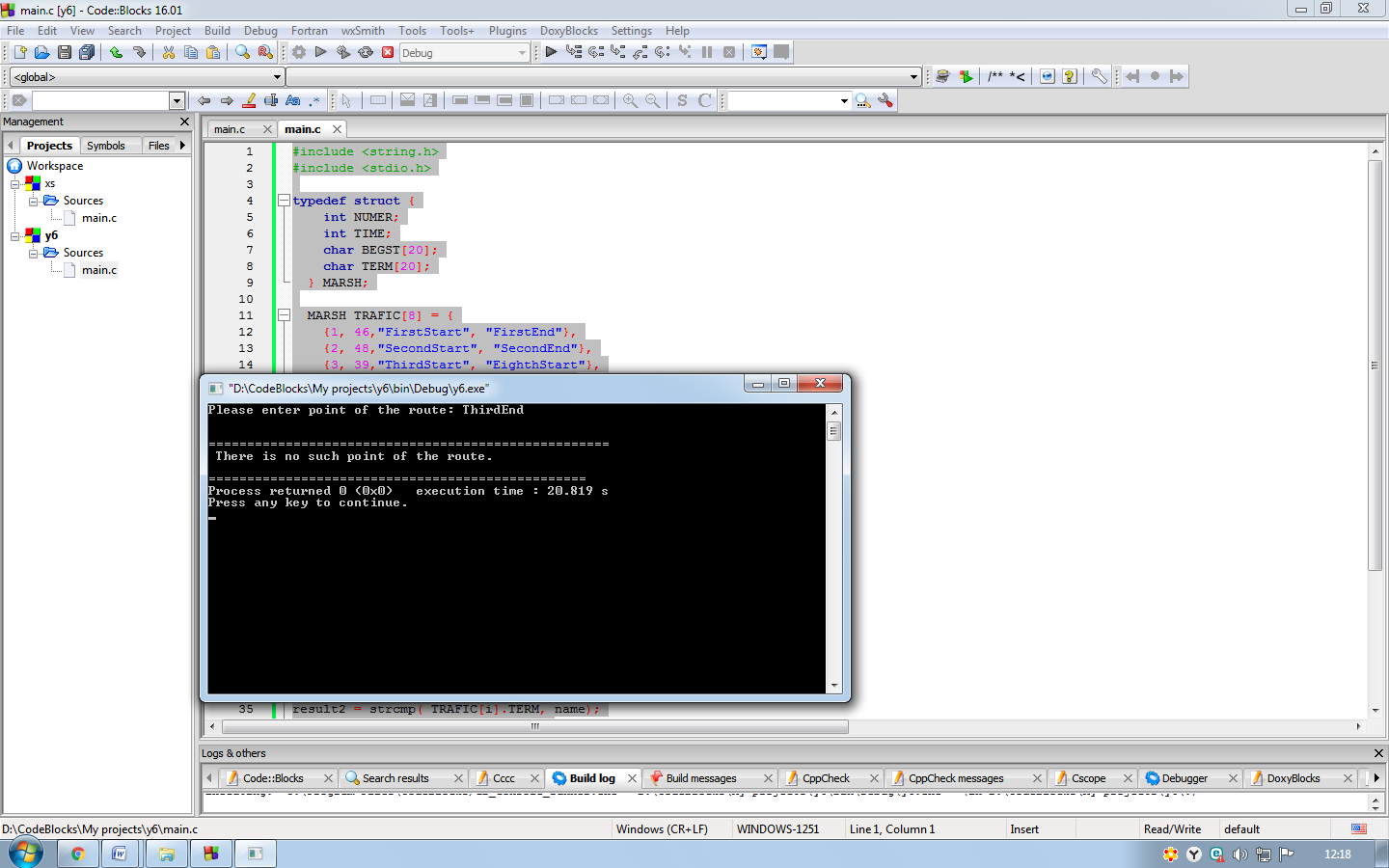
return 0;

}

Пример выполнения программы







**Индивидуальное задание №4**

Задание: *Написать программу для сортировки одномерного массива символов по убыванию методом бинарных вставок. Размерность массива, а также границы диапазона изменения элементов N и M вводятся пользователем с клавиатуры, а сам массив заполняется случайными числами из указанного пользователем диапазона. Сортировку массива и генерацию массива оформить в виде отдельных функций.*

Блок-схема

Начало

Ввод

N

Ввод

a

Ввод

b

massivei:=random(b-a+1)+a

i:=1,N;1

Вывод

massivei

**А**

Ввод количества сортируемых элементов

Ввод начальной границы генерации

числа

Ввод конечной границы генерации

числа

Заполнение массива случайными элементами из отрезка[a;b]

Конец

**A**

massivei:=bin\_ins(massivei,N)

i:=N,1;-1

Вывод

massivei

Сортировка элементов массива по возрастанию методом бинарных вставок

Вывод отсортированного массива

massivei-1>

massivei

Начало функции

bin\_ins

i:=1,N;1

x:=massivei

left:=0

right:=i-1

mid:=

massivemid<x

да

left:=mid+1:

right:=mid-1

нет

left<=right

j:=i-1, j>=left;-1

massivej+1:=

massivej

Конец функции

bin\_ins

Описание программы**:**

N –количество подлежащих сортировке элементов;

a – начальная граница генерации числа;

b –конечная граница генерации числа;

massive[] –массив, куда записываются генерируемые числа;

i,j –переменный циклов и индексы массива;

gen(massive,a,b,N) –функция, заполняющая массив случайными числами из отрезка[a;b];

bin\_ins(massive, N) –функция по сортировке массива по возрастанию методом бинарных вставок;

a –указатель на сортируемый массив;

x –элемент на правильном месте(он включается в отсортированную часть);

left –индекс - левая граница отсортированной части;

right –индекс - правая граница отсортированной части;

mid –индекс - середина последовательности;

Исходный код программы

#include <stdlib.h>

#include <stdio.h>

#include <math.h>

#include <locale.h>

void gen(char \*massive, int a, int b, int N) // Генерация массива

{ int i;

for (i = 0; i < N; i++)

{

massive[i] = a + rand() % (b - a + 1);

}

}

void bin\_ins(char \*a, int n) // Сортировка бинарными вставками

{

int x,left,right,mid;

int i,j;

for (i=1; i<n; i++)

{

if (a[i-1]>a[i])

{

x=a[i]; // x – включаемый элемент

left=0; // левая граница отсортированной части массива

right=i-1; // правая граница отсортированной части массива

do {

mid = (left+right)/2; // mid – новая "середина" последовательности

if (a[mid]<x ) {left= mid+1;}

else {right=mid-1;}

} while (left<=right); // поиск ведется до тех пор, пока левая граница не окажется правее правой границы

for (j=i-1; j>=left;j--) {a[j+1]= a[j];}

a[left]= x;

}

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int N, a, b;

printf("Введите размер массива: ");

scanf("%d", &N);

printf("Введите начальную границу генерации числа: ");

scanf("%d", &a);

printf("Введите конечную границу генерации числа: ");

scanf("%d", &b);

char \*massive;

massive = (char\*)malloc(N \* sizeof(int));

gen(massive, a, b, N);

printf("\n Исходный массив: \n");

int i;

for (i = 0; i < N; i++)

printf("%d ", massive[i]);

bin\_ins(massive, N);

printf("\n");

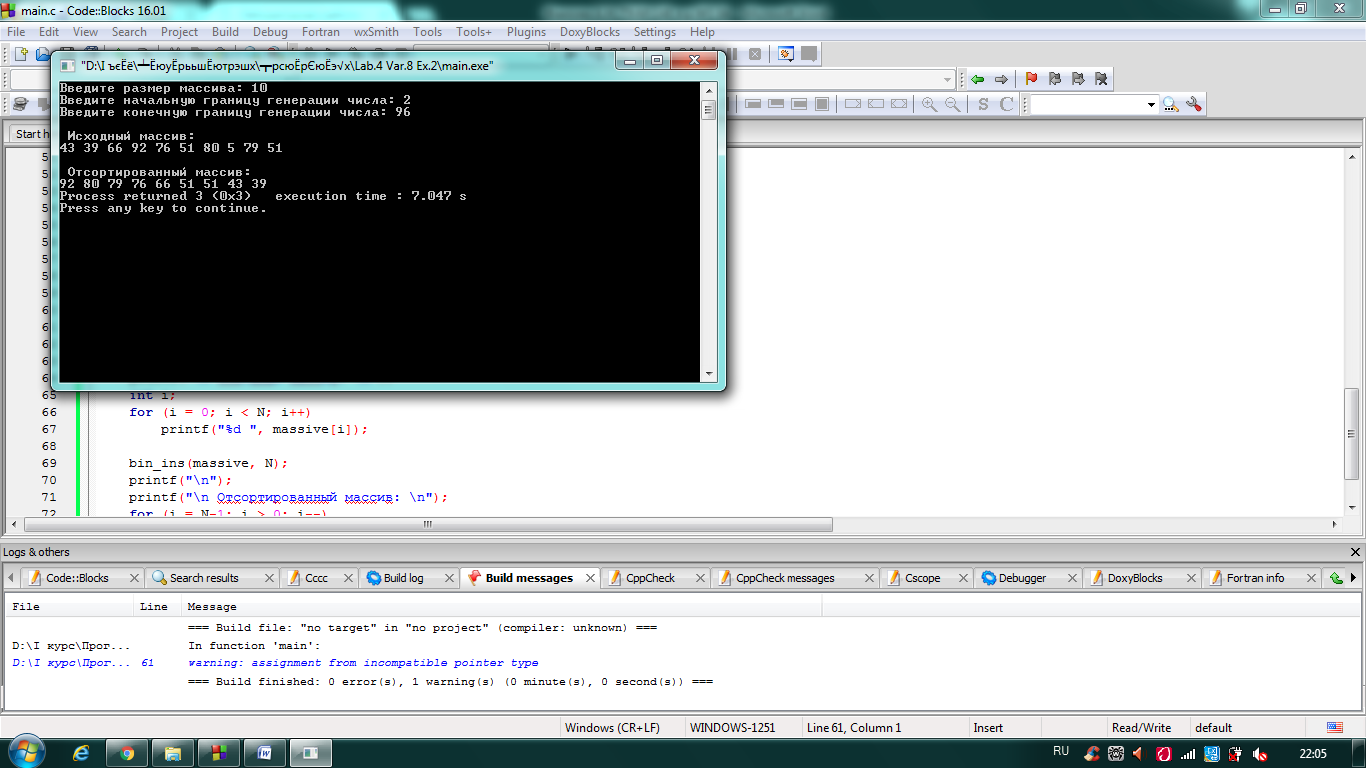
printf("\n Отсортированный массив: \n");

for (i = N-1; i > 0; i--)

printf("%d ", massive[i]);

}

Пример выполнения программы



**Вывод**

На данной лабораторной работе мы изучили принципы работы с указателями и структурными типами данных (массивы и структуры) в СИ; получили практические навыки в реализации алгоритмов сортировки одномерных массивов.

**Список использованной литературы**

1. В.В. Подбельский, С.С.Фомин. Программирование на языке Си. М.: ФиС, 1999, 600 с.
2. Функции [Электронный ресурс] / Бикмеев А. Т. //

<http://bikmeyev-at.ugatu.su/students/CPP/Lab102/index02.html> (Дата обращения 10.11.2017).

1. Керниган Б., Ритчи Д. Язык программирования Си. Пер. с англ., 3-е изд., испр. — СПб.: "Невский Диалект", 2001. - 352 с.