Филиал «Котельники» государственного бюджетного

образовательного учреждения высшего образования

Московской области «Университет «Дубна»»

**Пояснительная записка**

**по курсовой работе по дисциплине**

**«Программирование на языке высокого уровня»**

**ВАРИАНТ №7**

Выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

студент группы ИВТ-11: Каткова Е.А.

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доцент, к.т.н.: Артамонов Ю.Н.

Котельники – 2019

**Содержание**

[**Введение** 3](#_Toc9484174)

[**Глава I . РАЗРАБОТКА ЧИСЛЕННЫХ АЛГОРИТМОВ** 4](#_Toc9484175)

**1.1** [**Суммирование рядов и вычисление элементарных функций** 4](#_Toc9484176)

**1.2** [**Приближенные методы нахождения корней уравнения** 9](#_Toc9484177)

**1.2.1** [**Метод касательных** 9](#_Toc9484178)

**1.2.2** [**Метод секущих** 15](#_Toc9484179)

[**Глава II . РАЗРАБОТКА ИГРОВОЙ ПРОГРАММЫ** 21](#_Toc9484180)

[**Заключение** 32](#_Toc9484181)

[**Список литературы** 32](#_Toc9484182)

**Введение**

C — это достаточно "древний" язык программирования, он сформировался в начале 70-х. Не смотря на это, C — живой язык в том смысле, что он активно применяется в настоящее время. Он был придуман, использовался и используется для написания существенных частей программного кода Unix-подобных операционных систем. Также на нем пишут утилиты, компиляторы и реже прикладные программы. Поэтому C называют системным языком программирования.

Его живучесть можно объяснить тем, что принципы работы операционных систем относительно универсальны, они не подвержены тому прогрессу и разнообразию, которые можно наблюдать в среде десктопного и мобильного ПО, Web-приложений. C не является языком достаточно высокого уровня, он ближе к архитектуре компьютера. В результате программы на C получаются компактными и быстрыми.

C не поддерживает объектно-ориентированного программирования. Поддержка ООП реализована в C++. Хотя последний возник на основе языка C, он не является его "продолжением", а представляет собой отдельный язык, который можно изучать, не зная C. Однако изучение C полезно перед знакомством с его "продвинутым младшим братом", т.к. синтаксис языков похож, C не перегружает мозг начинающего программиста сверх возможностей и приучает к пониманию сути происходящего.

Подходит ли C для первого знакомства с программированием? Если вы не обучаетесь в вузе по специальности, связанной с IT, то нет. C предполагает понимание организации и принципов работы аппаратного обеспечения, в частности – памяти. Здесь многое делается с помощью указателей, они играют ключевую роль; эта тема достаточно сложная для понимания и обычно не изучается в школе.

Конечно, с помощью C можно изучать основы программирования и не изучать при этом указатели. Однако человек будет думать, что знает C, мало зная о нем, по существу. Язык C был создан разбирающимися в устройстве «железа» профессиональными программистами для написания операционной системы UNIX. Его не задумывали как язык для обучения начинающих.

**Глава I. Суммирование ядов и вычисление элементарных функций**

Ряд Маклорена — это степенной ряд, в котором слагаемыми служат действительная функция f(x) в точке 0 и её производные всех порядков в точке 0, делённые на факториал, соответствующий порядку производной и умноженные на x в соответствующей степени.

Формула

f(x)=f(0)+f′(0)1!x+f′′(0)2!x2+f′′′(0)3!x3+…+f(n)(0)n!xn+…

f(x)=∑n=0∞f(n)(0)n!xn

Ряд Маклорена является частным случаем ряда Тейлора при a=0.

Примеры

Разложение «элементарных» функций в ряд Маклорена:

ex=1+x1!+x22!+x33!+…+xnn!+…, ∀x∈(−∞;+∞)

ln(1+x)=x−x22+x33−x44+…+(−1)n+1xnn+…, ∀x∈(−1;1]

sinx=x−x33!+x55!−x77!+…+(−1)nx2n+1(2n+1)!+…, ∀x∈(−∞;+∞)

cosx=1−x22!+x44!−x66!+…+(−1)nx2n(2n)!+…, ∀x∈(−∞;+∞)

arcsinx=x+x32!!⋅3+3!!x54!!⋅5+5!!x76!!⋅7+…+(2n−1)!!x2n+1(2n)!!(2n+1)+…, ∀x∈(−1;1)

arctgx=x−x33+x55−x77+…+(−1)nx2n+12n+1+…, ∀x∈(−1;1)

shx=x+x33!+x55!+x77!+…+x2n+1(2n+1)!+…, ∀x∈(−∞;+∞)

chx=1+x22!+x44!+x66!+…+x2n(2n)!+…, ∀x∈(−∞;+∞)

arshx=x−x32!!⋅3+3!!x54!!⋅5−5!!x76!!⋅7+…+(−1)n(2n−1)!!x2n+1(2n)!!(2n+1)+…, ∀x∈(−1;1)

arcthx=x+x33+x55+x77+…+x2n+12n+1+…, ∀x∈(−1;1)

Найти:

Реализуйте вычисление по этому представлению, и проверьте насколько быстро сходится процесс вычислений (т.е. сколько слагаемых надо взять для получения результата с заданной точностью при различных значениях х). Полученные результаты расчета в виде таблицы в пояснительной записке: значение х, точность, количество членов ряда.

Решение:

Исходный код программы по методу касательных реализован в сравнительном написании с методом секущих. за основу директивы препроцессора #define взяты значения eps с точностью до 0.000001 и значение числа е 2.71828. для определения выбора, какое уравнения будет исчисляться тем или иным методом, использован оператор switch (numberOfEquation), входные данные которого передаются по ссылке из условий по каждому методу. если условие функции выполняется, мы возвращаем результат самой функции с применением искомых координат. основная функция int main() выводит на экран результаты вычислений с заданной точностью, приравнивая их к уравнениям.

#include <stdio.h> // подключение стандартной библиотеки, файл ввода-вывода

#include <stdlib.h> // подключение стандартной библиотеки, для работы с функциями

#include <math.h> // подключение математической библиотеки

double f(double ); // прототип функции по нахождению левой части представления

double f(double x) // функция по нахождению левой части представления

{

//return 1/4\*log((1+x)/(1-x))+1/2\*atanf(x); в таком порядке не вычисляет

return log((1+x)\*1.0/(1-x))/4+atan(x)/2; // функция возвращает результат

}

double row(double ); // прототип функции по нахождению правой части представления

double row(double x) // функция по нахождению правой части представления

{

int count=0; // переменная, которая считает кол-во слагаемых

double n=0,s=0,eps=1e-14; // переменные вещественного типа

{

while(abs(f(x))-abs(s)>eps) // условием выхода из цикла является выражение,

// которое считает разницу между правой и левой сторонами в соответствии с заданной точностью

{

s+=pow(x,4\*n+1)/(4\*n+1.0); // считает сумму всех слагаемых согласно условию

n++; // перемнная увеличивается на 1

count++; // перемнная для подсчета кол-ва слагаемых

}

printf("\nКол-во слагаемых = %d\n",count); // вывод кол-ва слагаемых

return s; // функция возвращает результат в

}

}

int main () // функция начала работы программы

{

double n; // переменная вещественного типа

printf("Введите значение x = "); // ввод значения

scanf ("%lf",&n); // считывает введенное значение

while(1) // цикл на проверку ввода корректных значений

{

if(n>-1 && n<1) // условие, при котором представление выполняется

{

printf("\n%.15lf = %.15lf\n", f(n),row(n)); // вывод равенства

break; // выход из цикла после вывода равенства

}

else // условие, при котором представление не выполняется

{

printf("Введите x в пределах от -1 до 1\nx = "); // неправильный ввод

// требуется ввести значение еще раз согласно условию

scanf ("%lf",&n); // считывает введенное значение

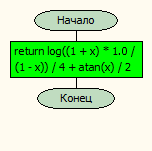
}

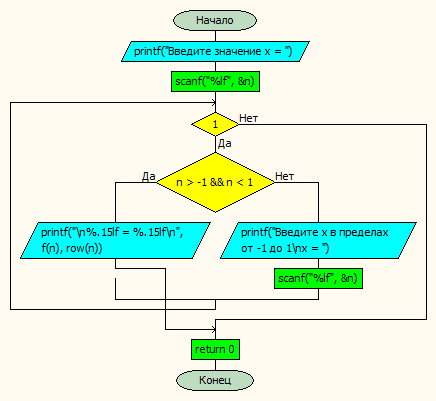
}

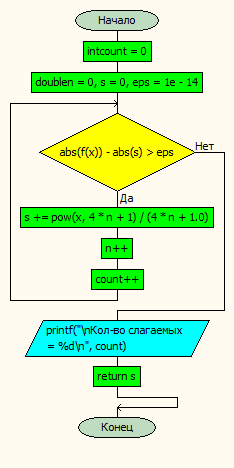
return 0; // функция возвращает число

}

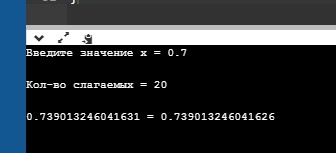
А теперь рассмотрим блок-схемы:





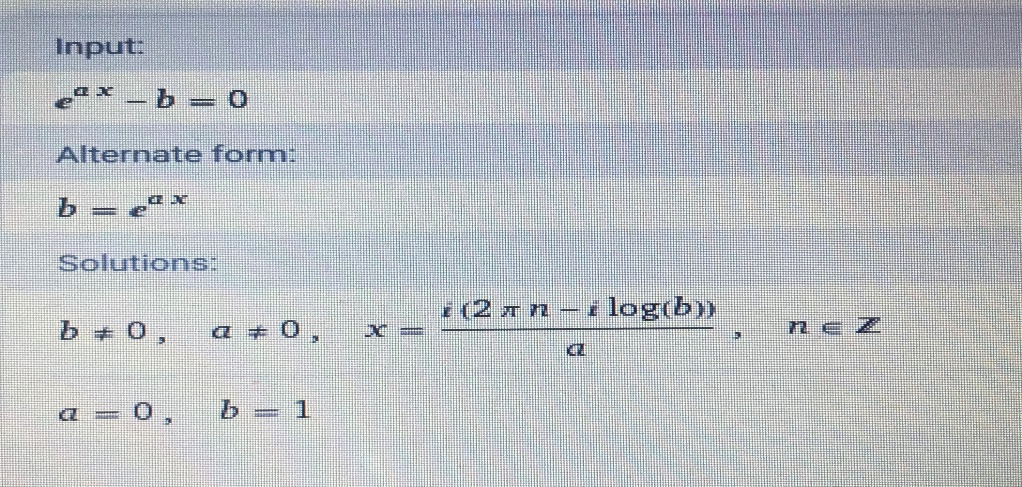


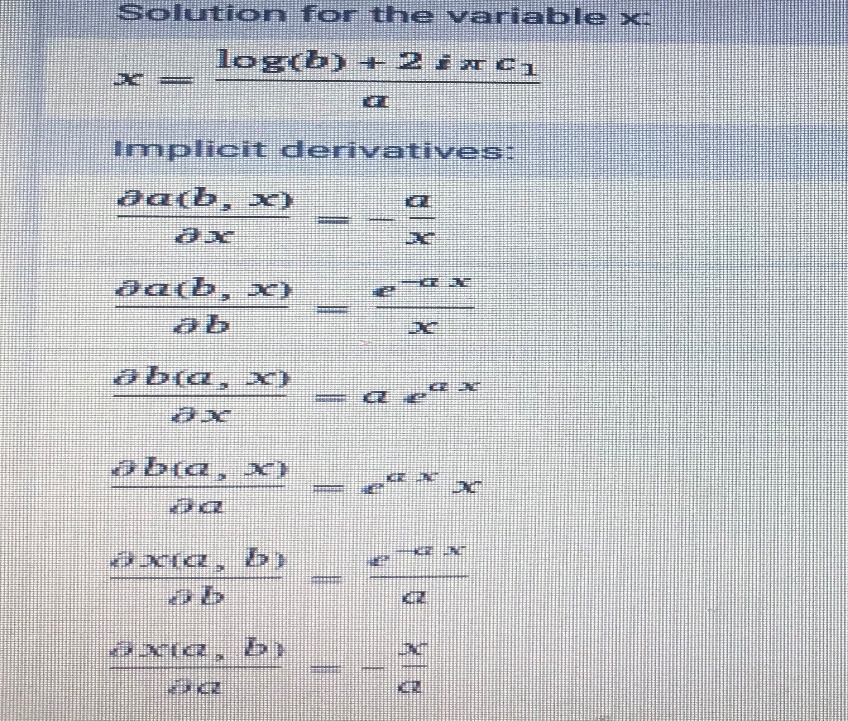
|  |  |
| --- | --- |
| Значение х | 0.7 |
| Кол-во слагаемых | 20 |
| Полученное значение | 0.739013246041631 |



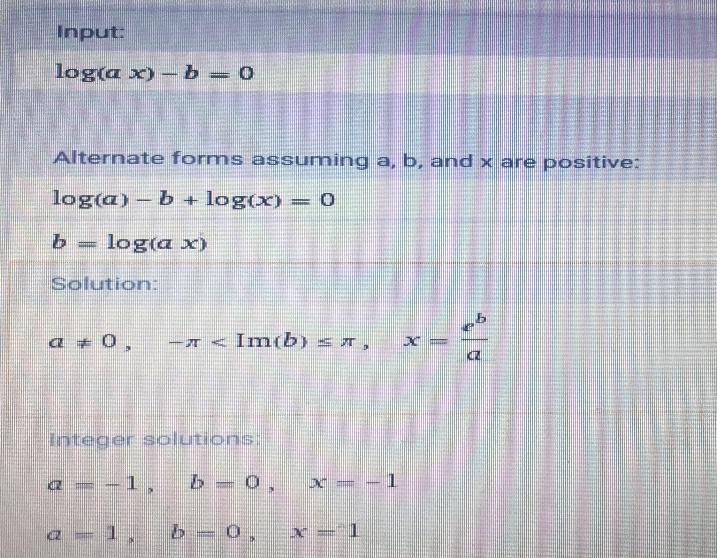
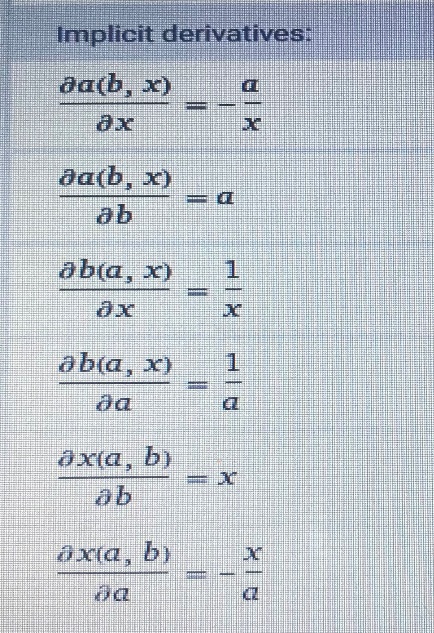
В заключении рассмотрим Wolfram Alfa :

2.

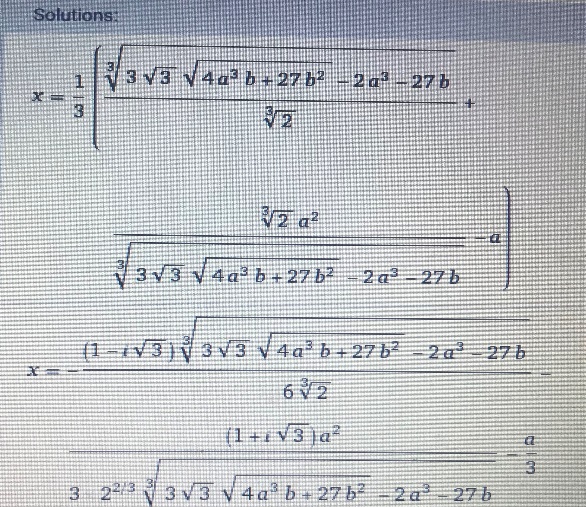
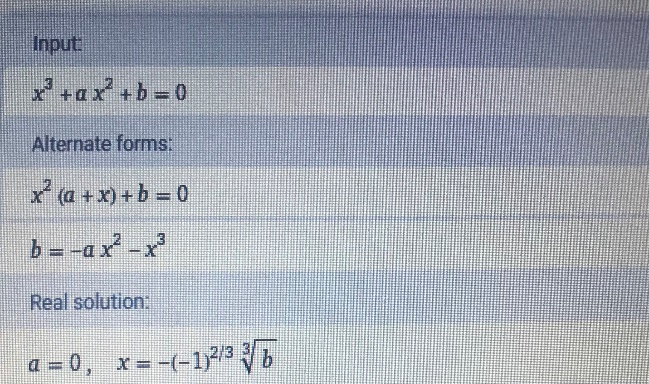


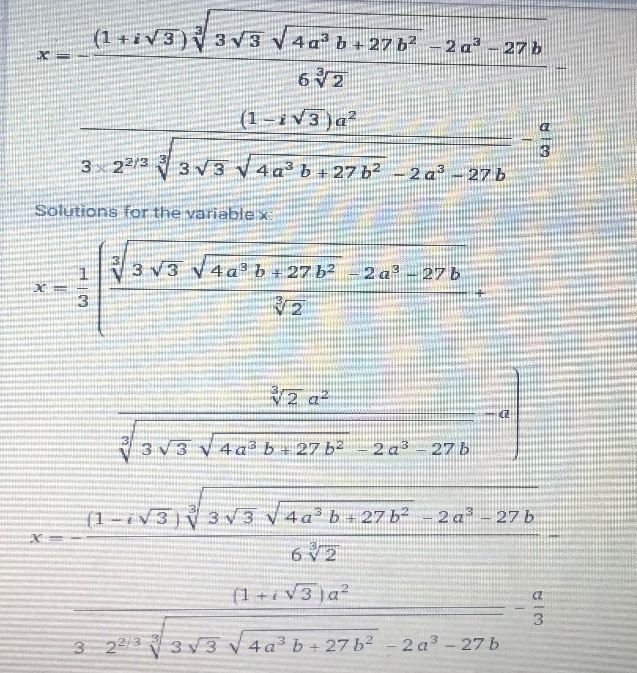
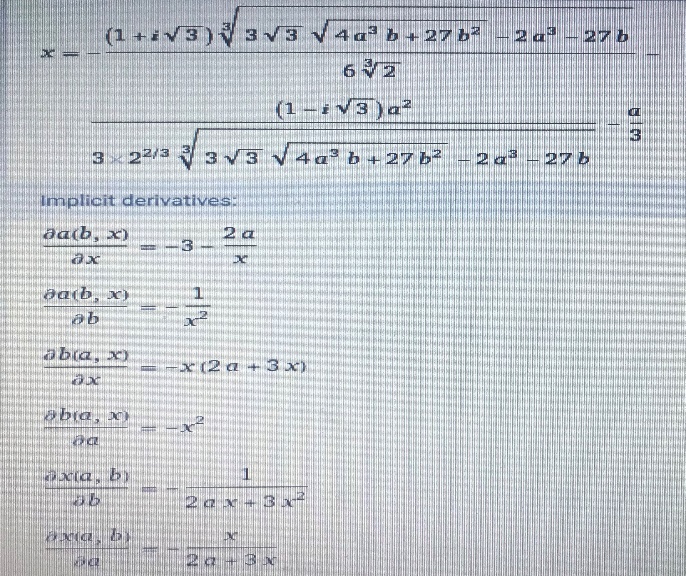


3.

4.



Заключение: В данном разделе мы рассмотрели принцип работы рядов Маклорена. Одним из самых замечательных его результатов является эта формула, в соответствии с которой в слагаемыми служат действительная функция f(x) в точке 0 и её производные всех порядков в точке 0, делённые на факториал, соответствующий порядку производной и умноженные на x в соответствующей степени.

**Глава II. Приближенные методы нахождения корней уравнения**

Проблемы решения каких-либо уравнений часто возникают на практике. Например, в экономике, когда развивается бизнес, вам необходимо определить, при каких условиях доход будет равен определенной величине, в медицине, при изучении воздействия лекарств, необходимо узнать, в каком случае концентрация вещества будет доходить до необходимо уровня и т.д.

**Решить уравнение** — значит найти все его корни, то есть те значения x, которые обращают уравнение в тождество.

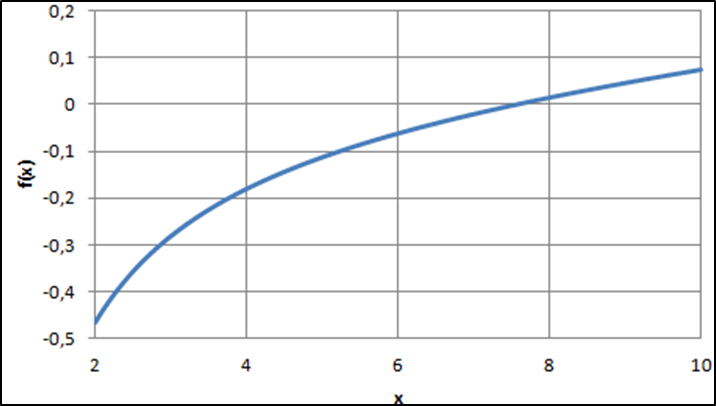
Для примера рассмотрим задачу решения уравнения

sin(x)=1/x

Где угол x задан в градусах. Указанное уравнение можно переписать в виде

sin(π/(180°) x)-1/x=0

Для графического отсечения корней достаточно построить график функции



Из рисунка видно, что корень нашего уравнения лежит в промежутке

x∈(6;8)

**Метод секущих** - получается из метода касательных заменой f'(x^k) разностным приближением:

f'(x^k) \approx \frac{f(x^k)-f(x^{k-1})}{x^k-x^{k-1}}.

В результате получим формулу итерационного процесса:

x^{k+1}=x^k-\frac{x^k-x^{k-1}}{f(x^k)-f(x^{k-1})}f(x^k), \;k=1,2,\ldots

Метод секущих является двухшаговым, то есть новое приближение x^{k+1} определяется двумя предыдущими итерациями x^k и x^{k-1}. В методе (1) необходимо задавать два начальных приближения x^0 и x^1.

Скорость сходимости метода будет линейной: |x^{k+1}-x\*|=O(k^k-x\*).

**Метод деления пополам** - этот метод является особенно надёжным при нахождении корня уравнения f(x)=0, когда о поведении функции f(x) мало, что известно. Пусть f(x) – функция действительной переменной x и пусть известен интервал (a; b), на котором функция меняет знак. Следовательно, между a и b существует точка, в которой функция обращается в ноль. Если разделить интервал пополам и определить, положительна или отрицательна функция в точке деления, то тем самым найдем под интервал, в котором функция меняет знак.

При заданной абсолютной погрешности Е алгоритм метода деления пополам состоит из следующих шагов:

1) вычислить f(a) и f(b)

2) положить c=(a+b)/2, вычислить f(c)

3) если f(c) = f(a), то заменить a на с, в противном случае заменить b на с

4) если b-a>E, перейти к шагу 2; в противном случае прекратить вычисления, поскольку достигнута требуемая точность.

**Рассмотрим уравнения, корни которых, нам необходимо найти.**

Дано:

2) e^ax-b=0

3) log2(ax)-b=0

4) x^3+ax^2+b=0

Найти: Провести тестирование методов деления пополам и секущих в соответствии с вариантом на примере решения уравнений:

Решение: Исходный код программы по методу касательных реализован в сравнительном написании с методом секущих. за основу директивы препроцессора #define взяты значения eps с точностью до 0.000001 и значение числа е 2.71828. для определения выбора, какое уравнения будет исчисляться тем или иным методом, использован оператор switch (numberOfEquation), входные данные которого передаются по ссылке из условий по каждому методу. если условие функции выполняется, мы возвращаем результат самой функции с применением искомых координат. основная функция int main() выводит на экран результаты вычислений с заданной точностью, приравнивая их к уравнениям.

// Метод деления отрезка пополам и метод секущих

\*/

// Constants and libs

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <stdio.h>

#include <math.h>

#include <cmath>

using namespace std;

// Точность

#define eps 0.0000001

// задаем тип func

typedef double (\*func)(double x, double a, double b, short int numberOfEquation);

// прототип вычисляемой функции

double fx(double, double, double, short int);

double Log2(double n)

{

// log(n)/log(2) is log2.

return log(n) / log(2);

}

// Вычисляемые функции

double fx(double x, double a, double b, short int numberOfEquation) {

switch (numberOfEquation) {

case 0: return pow(M\_E, a \* x) - b;

case 1: return Log2(a \* x) - b;

case 2: return pow(x, 3) + a \* pow(x, 2) + b;

}

return 0;

}

// Метод секущих

double\* decision\_secant\_method(func fx, double x1, double x2, double a, double b) {

double\* answers = new double[3];

double x1\_OriginValue = x1;

double x2\_OriginValue = x2;

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

double v;

x1 = x1\_OriginValue;

x2 = x2\_OriginValue;

// пока не достигнута точность eps

while (fabs(x1 - x2) > eps) {

v = (x1 \* fx(x2, a, b, i) - x2 \* fx(x1, a, b, i)) / (fx(x2, a, b, i) - fx(x1, a, b, i));

x1 = x2;

x2 = v;

}

answers[i] = x1;

}

return answers;

}

// Метод деления отрезков

double decision\_line\_division\_method(double x1, double x2, double a, double b, short int NumOfEquation) {

double ksi, returnValue;

ksi = (x1 + x2) / 2.0;

if ((fabs(fx(x1, a, b, NumOfEquation) - fx(x2, a, b, NumOfEquation)) <= eps) || (fabs(fx(ksi, a, b, NumOfEquation)) <= eps)) {

returnValue = (x1 + x2) / 2.0;

return returnValue;

}

if (fx(x1, a, b, NumOfEquation) \* fx(ksi, a, b, NumOfEquation) <= 0.0) return decision\_line\_division\_method(x1, ksi, a, b, NumOfEquation);

else return decision\_line\_division\_method(ksi, x2, a, b, NumOfEquation);

}

int main()

{

setlocale(LC\_ALL, "Russian");

double a = 3, b = 1;

double x1 = 0, x2 = 5;

double\* answer\_Secant\_method = decision\_secant\_method(fx, x1, x2, a, b);

printf("interval:[%.2lf,%.2lf] \n", x1, x2);

cout << "a = " << a << ", b =" << b << "\n \n";

cout << "Результаты для функции 'Метод секущих':\n";

cout << "\n e^ax - b = 0 | " << answer\_Secant\_method[0]

<< "\n log2(ax) - b = 0 | " << answer\_Secant\_method[1]

<< "\n x^3 + ax^2 + b = 0 | " << answer\_Secant\_method[2]

<< " \n \n \n";

double answer\_Line\_division\_method\_0 = decision\_line\_division\_method(x1, x2, a, b, 0);

double answer\_Line\_division\_method\_1 = decision\_line\_division\_method(x1, x2, a, b, 1);

double answer\_Line\_division\_method\_2 = decision\_line\_division\_method(x1, x2, a, b, 2);

cout << "Результаты для функции 'Метод деления отрезка':\n";

cout << "\n e^ax - b = 0 | " << answer\_Line\_division\_method\_0

<< "\n log2(ax) - b = 0 | " << answer\_Line\_division\_method\_1

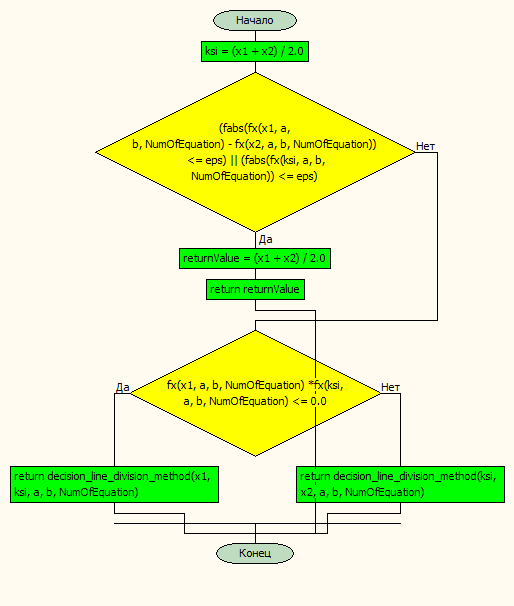
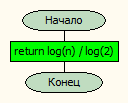
<< "\n x^3 + ax^2 + b = 0 | " << answer\_Line\_division\_method\_2

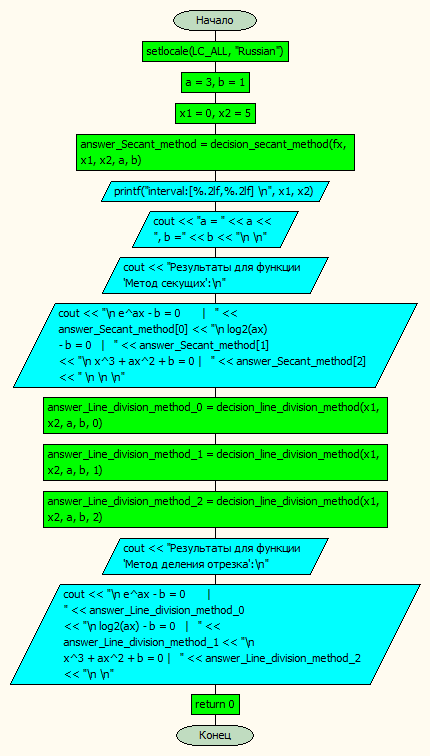
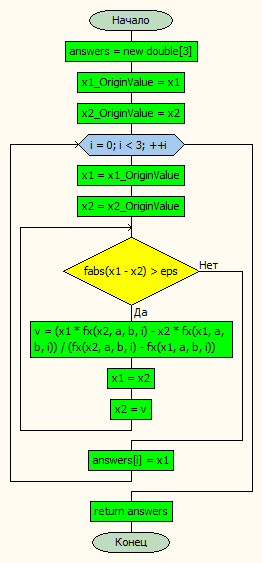
<< "\n \n";

return 0;

}

А также рассмотрим блок-схему данного программного кода, на примере уравнения.

1. 



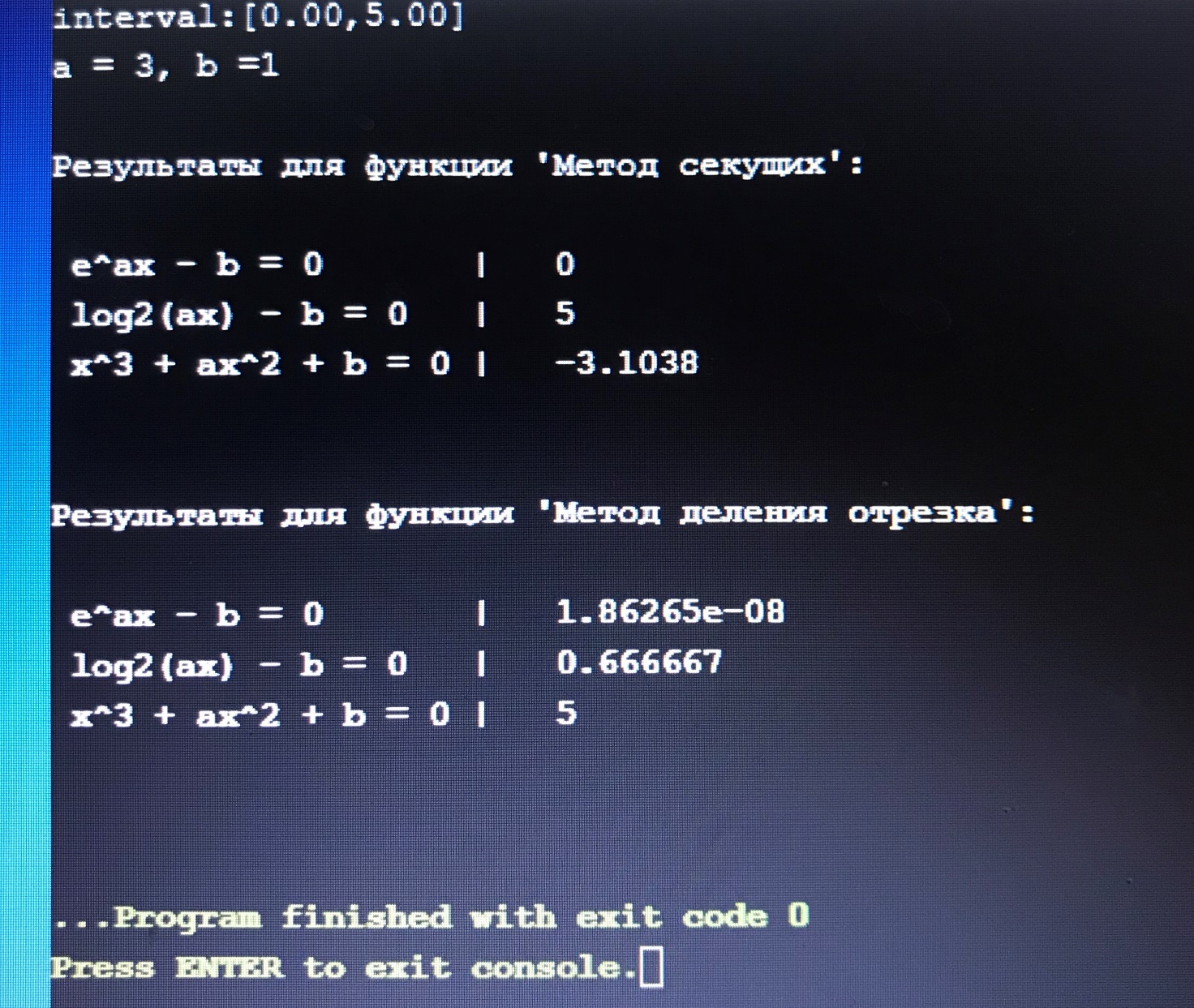
Рассмотрим таблицу данных для уравнения кода:

Метод секущих

|  |  |
| --- | --- |
| Уравнение | Полученное значение |
| e^ax – b = 0 | 0 |
| log2 (ax) – b = 0 | 5 |
| x^3 + ax^2 + b = 0 | -3.1038 |

Метод деления пополам

|  |  |
| --- | --- |
| Уравнение | Полученное значение |
| e^ax – b = 0 | 1.86265e-08 |
| Log2(ax) – b = 0 | 0.666667 |
| x^3 + ax^2 + b = 0 | 5 |



Заключение: в данном разделе мы рассмотрели метод секущих и метод деления, а также научились находить корни с помощью данных методов. Исходя из результатов, можно сделать вывод о том, что тригонометрические функции требуют больших итераций для точных вычислений.

**Глава III. Разработка игровой программы**

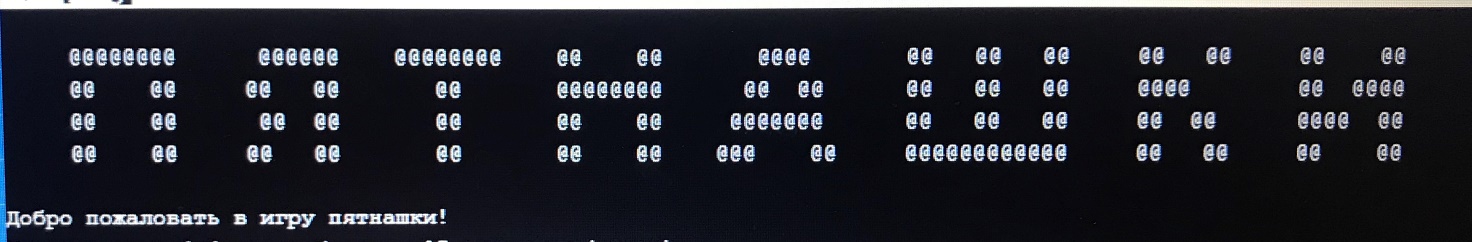
В данной главе мы разберем создание игры на языке С. Написание игр является креативным делом, и у каждого одна и та же игра может быть написана по-разному. Мне попалась игра под названием «Пятнашки». Задача всей игры состоит в том, чтобы путем перемещения фишек получить последовательный ряд чисел.

Имеется поле 4х4 для набора из 15 элементов(чисел), соответственно на поле остаётся незаполненным один элемент.

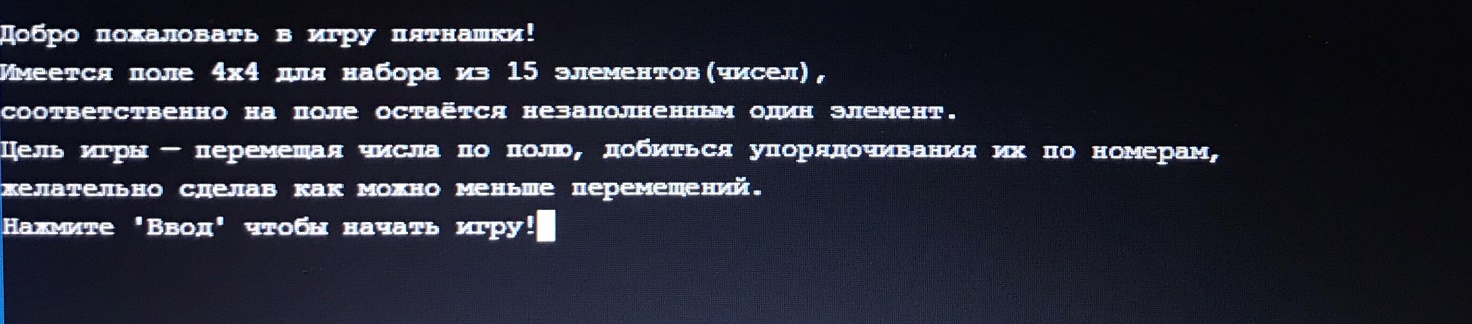
Цель игры — перемещая числа по полю, добиться упорядочивания их по номерам, желательно сделав как можно меньше перемещений.

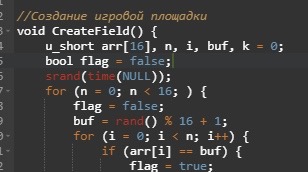
Теперь мы можем перейти к коду.

Игра приветствует нас красивым меню.



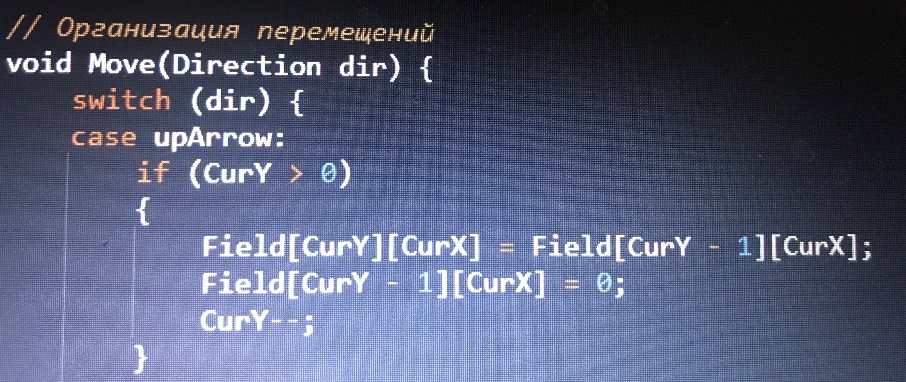
В самом начале мы можем прочитать правила игры, а также начать игру.



При создании игровой площадки для определения входных значений в массив использована генерация случайных чисел.

При заполнении массива случайными числами объявлена логическая переменная типа bool, которая при невыполнении условия на true переходит к реализации вывода игровой площадки на экран.

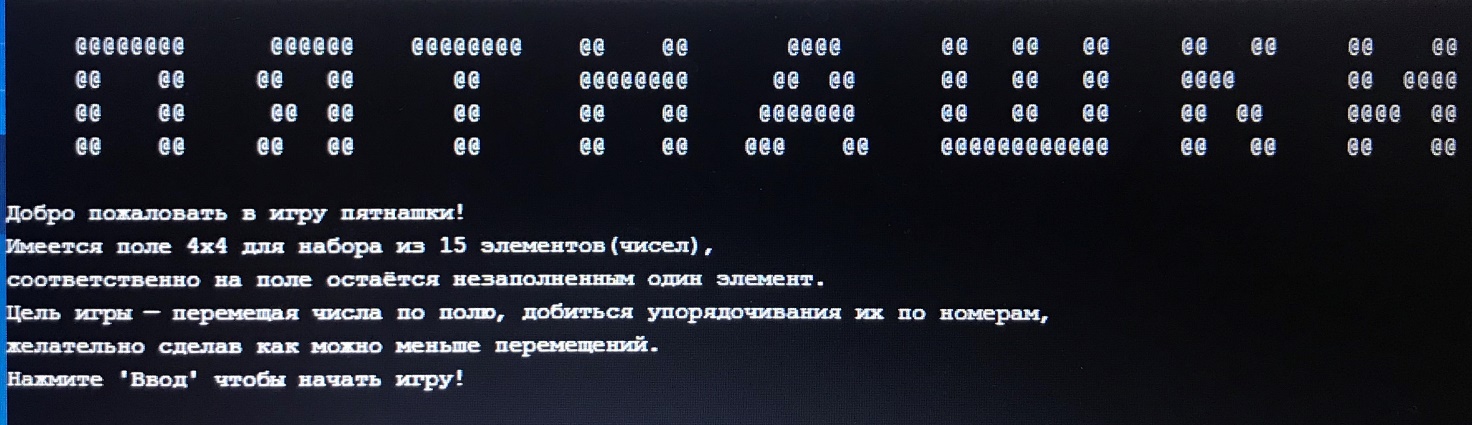
Реализация выбора при организации перемещений основана на операторе switch для большей эффективности кода.



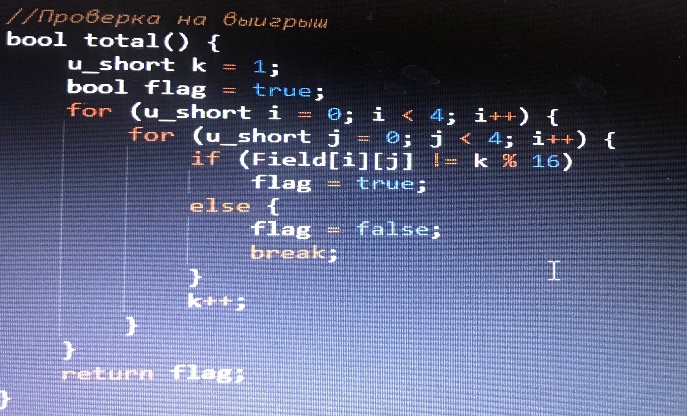
Перемещение по игровой площадке происходят согласно условиям входных значений. Если клетка содержит значение больше нуля, то происходит перемещение. Ей присваивается значение сверху, а та клетка чьё значение было заимствовано, обнуляется.

Когда условие выполнения содержит значение переменной меньше трех, на позицию определяется нижнее значение игровой площадки. Аналогичные условия реализованы для координат, меняющих положение слева и справа.

Главная функция int main() встречает игрока выводом приветствия на экран, и ждет дальнейших действий, когда игрок будет готов, нажав enter, с помощь макроса getchar(). После соглашения игрока на дальнейший этап прохождения игры реализуется вывод игровой площадки на экран, вызвав функцию void coutArr().



Ко всему прочему в функции int main() содержится цикл, который проверяет игрока на выигрыш. В нем также, как и в функции void Move(Direction dir) использован оператор switch. По окончании проверки функция выводит сообщение об окончании игры и поздравляет игрока.



**Приложение**

// Project33.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#include<time.h>

#include<conio.h>

#include<iostream>

#include <limits>

#define KEY\_UP 72

#define KEY\_DOWN 80

#define KEY\_LEFT 75

#define KEY\_RIGHT 77

typedef unsigned short u\_short;

typedef unsigned int u\_int;

typedef enum Direction { upArrow, downArrow, rightArrow, leftArrow } Direction;

using namespace std;

u\_short Field[4][4];

u\_short CurX, CurY;

//Создание игровой площадки

void CreateField() {

u\_short arr[16], n, i, buf, k = 0;

bool flag = false;

srand(time(NULL));

for (n = 0; n < 16; ) {

flag = false;

buf = rand() % 16 + 1;

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

if (arr[i] == buf) {

flag = true;

break;

}

}

if (!flag) {

arr[n] = buf;

n++;

}

}

for (n = 0; n < 4; n++)

for (i = 0; i < 4; i++) {

Field[n][i] = arr[k];

k++;

}

Field[3][3] = 0;

CurX = 3; CurY = 3;

return;

}

// Организация перемещений

void Move(Direction dir) {

switch (dir) {

case upArrow:

if (CurY > 0)

{

Field[CurY][CurX] = Field[CurY - 1][CurX];

Field[CurY - 1][CurX] = 0;

CurY--;

}

else

printf("ВЫХОД ЗА ПРЕДЕЛЫ ПОЛЯ \n\n");

break;

case downArrow:

if (CurY < 3)

{

Field[CurY][CurX] = Field[CurY + 1][CurX];

Field[CurY + 1][CurX] = 0;

CurY++;

}

else

printf("ВЫХОД ЗА ПРЕДЕЛЫ ПОЛЯ \n\n");

break;

case rightArrow:

if (CurX < 3)

{

Field[CurY][CurX] = Field[CurY][CurX + 1];

Field[CurY][CurX + 1] = 0;

CurX++;

}

else

printf("ВЫХОД ЗА ПРЕДЕЛЫ ПОЛЯ \n\n");

break;

case leftArrow:

if (CurX > 0)

{

Field[CurY][CurX] = Field[CurY][CurX - 1];

Field[CurY][CurX - 1] = 0;

CurX--;

}

else

printf("ВЫХОД ЗА ПРЕДЕЛЫ ПОЛЯ \n\n");

break;

}

}

//ВЫВОД ПОЛЯ

void coutArr() {

system("cls");

printf("\n");

for (u\_int i = 0; i < 4; i++) {

for (u\_int j = 0; j < 4; j++) {

printf("\t%d ", Field[i][j]);

}

printf("\n\n");

}

return;

}

//Проверка на выигрыш

bool total() {

u\_short k = 1;

bool flag = true;

for (u\_short i = 0; i < 4; i++) {

for (u\_short j = 0; j < 4; i++) {

if (Field[i][j] != k % 16)

flag = true;

else {

flag = false;

break;

}

k++;

}

}

return flag;

}

int main() {

setlocale(LC\_ALL, "Russian");

printf("\n @@@@@@@@ @@@@@@ @@@@@@@@ @@ @@ @@@@ @@ @@ @@ @@ @@ @@ @@ \n");

printf(" @@ @@ @@ @@ @@ @@@@@@@@ @@ @@ @@ @@ @@ @@@@ @@ @@@@ \n");

printf(" @@ @@ @@ @@ @@ @@ @@ @@@@@@@ @@ @@ @@ @@ @@ @@@@ @@ \n");

printf(" @@ @@ @@ @@ @@ @@ @@ @@@ @@ @@@@@@@@@@@@ @@ @@ @@ @@ \n\n");

printf("Добро пожаловать в игру пятнашки!\n");

printf("Имеется поле 4х4 для набора из 15 элементов(чисел),\n");

printf("соответственно на поле остаётся незаполненным один элемент.\n");

printf("Цель игры — перемещая числа по полю, добиться упорядочивания их по номерам, \n");

printf("желательно сделав как можно меньше перемещений.\n");

cout << "Нажмите 'Ввод' чтобы начать игру!";

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

CreateField();

coutArr();

printf("Выберите направление - стрелочки 'вверх, вниз, влево, вправо'\n");

int c = 0;

while (!total()) {

c = 0;

switch ((c = getch())) {

case KEY\_UP:

Move(upArrow);

break;

case KEY\_DOWN:

Move(downArrow);

break;

case KEY\_LEFT:

Move(leftArrow);

break;

case KEY\_RIGHT:

Move(rightArrow);

break;

case 27:

printf("ХОРОШАЯ ПОПЫТКА! \n");

return 0;

break;

default:

break;

}

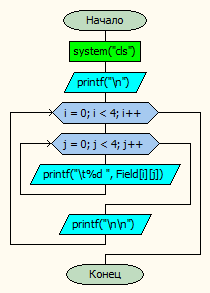
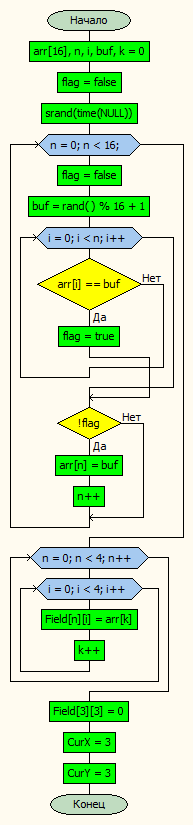
coutArr();

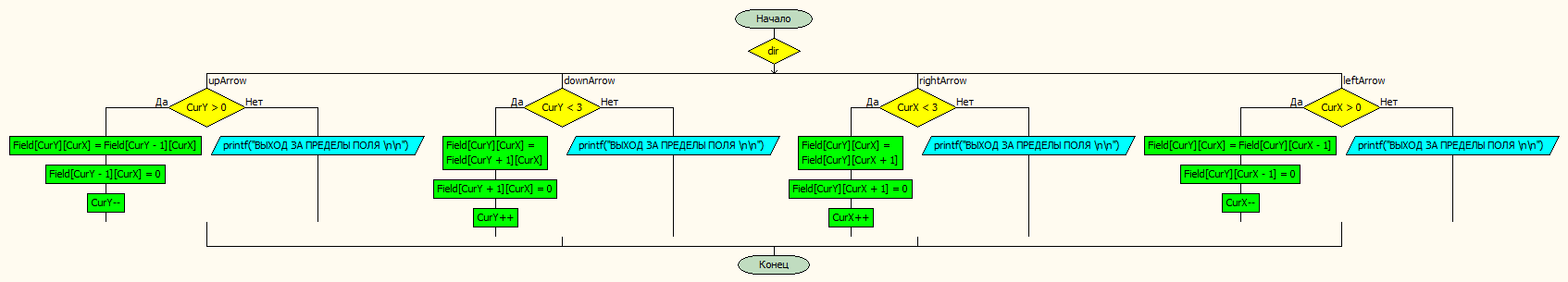
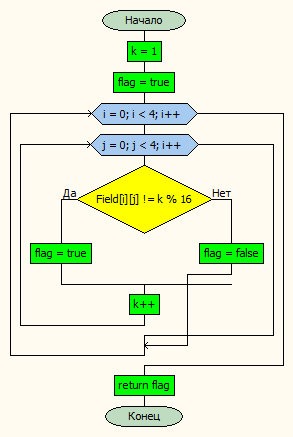
}

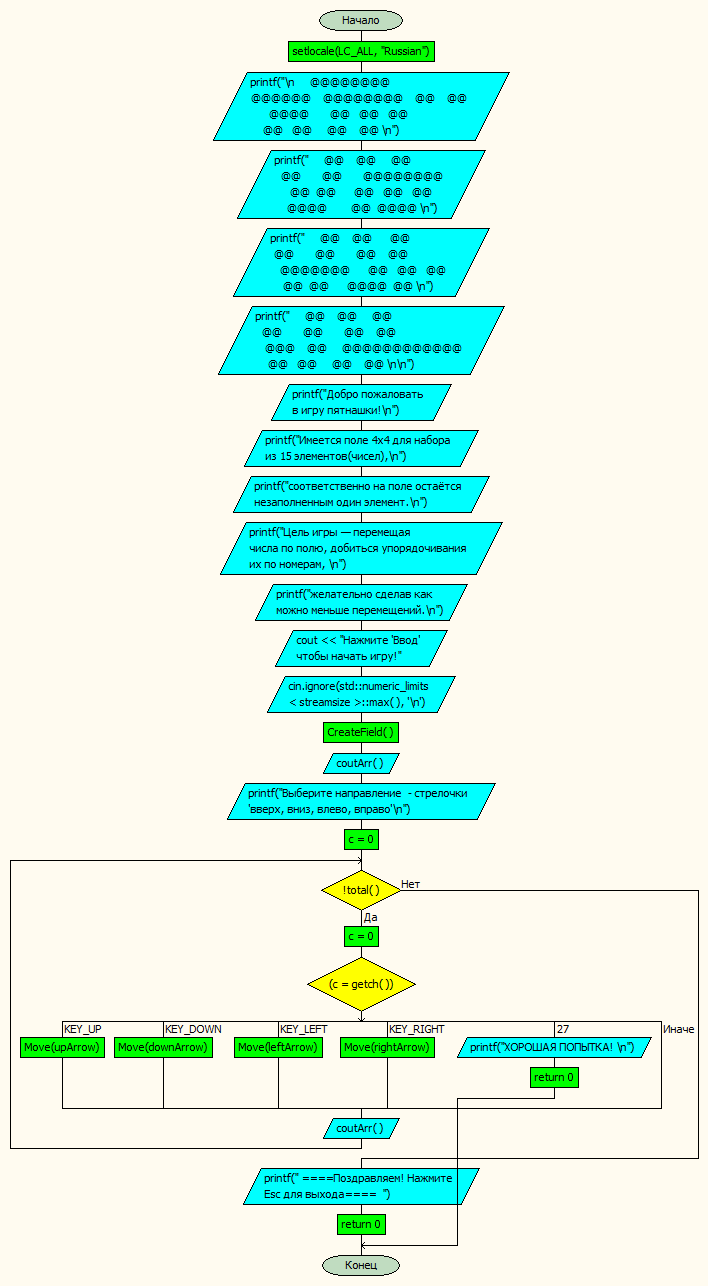
printf(" ====Поздравляем! Нажмите Esc для выхода==== ");

return 0;

}







**Заключение:** С это лучший язык для тех, то любит “свободу действий” в программировании. В данной курсовой работе мы в этом убедились. Благодаря С можно решить почти любое математическое выражение, найти корни уравнений, а также написать игру. Самым главным плюсом данного языка является его скорость. Не многие языки программированию могут похвастаться ею.

**Список литературы:**

[1] Юркин А. Задачник по программированию. – СПб.: Питер, 2002. – 192 с.

[1] Banahan, M. F. The C Book: Featuring the ANSI C Standard, 1988. – 268 c.

[2].Брайан У Язык программирования C. 1978. – 343 c.

[3] Стивен Скиена Алгоритмы. Руководство по разработке, 2011. – 719 c.

[4]Генри С.Уоррен мл. Алгоритмические трюки для программистов., 2019 – 512 c.

[5]Гай Я.Г - Справочное пособие по высшей математике. Том 1.,2018 – 272 c.

[6]Семакин И.Г Программирование, численные методы и математическое моделирование.,2017 - 299 с.

[7]Гателюк О.В Численные методы. Учебное пособие для академического бакалавриата.,2018 – 140 с.