МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

Курсовая работа
по дисциплине «Организация ЭВМ и систем»
ТЕМА: «ИССЛЕДОВАНИЕ ВНУТРЕННЕГО ПРЕДСТАВЛЕНИЯ РАЗЛИЧНЫХ ФОРМАТОВ ДАННЫХ»

Студенты гр. 1308 _____ Мельник Д. А.
Лепов А. В.

Преподаватель _____ Костичев С. В.

Санкт-Петербург
2022

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. В зависимости от номера варианта задания разработать алгоритм ввода с клавиатуры требуемых типов данных и показать на экране их внутреннее представление в двоичной системе счисления.

2. Написать и отладить программу на языке C++, реализующую разработанный алгоритм. Программа должна

- иметь дружественный интерфейс

- выводить на экран информативное сообщение при вводе некорректных данных

- предложить повторный ввод пока не будут введены корректные данные

3. В соответствии с заданием дополнить разработанный ранее алгоритм блоками для выполнения преобразования двоичного полученного кода исходного типа данных и последующего вывода преобразованного кода в двоичной системе счисления и в формате исходного данного.

Вариант 8.

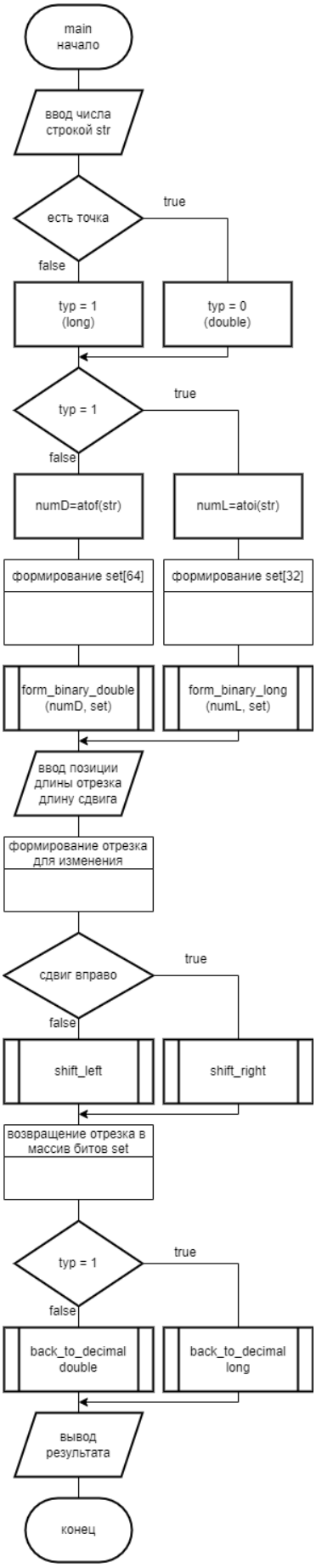
Выполнить циклический сдвиг в заданную пользователем сторону на заданное количество разрядов в пределах определённой группы разрядов, количество которых и номер старшего разряда в группе задаются с клавиатуры.

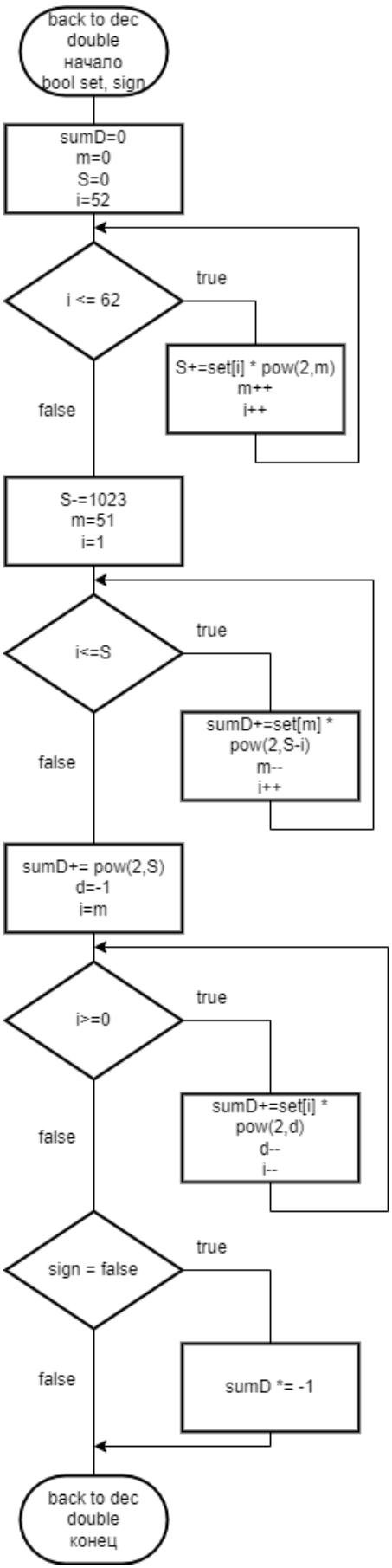
Распределение обязанностей в бригаде:

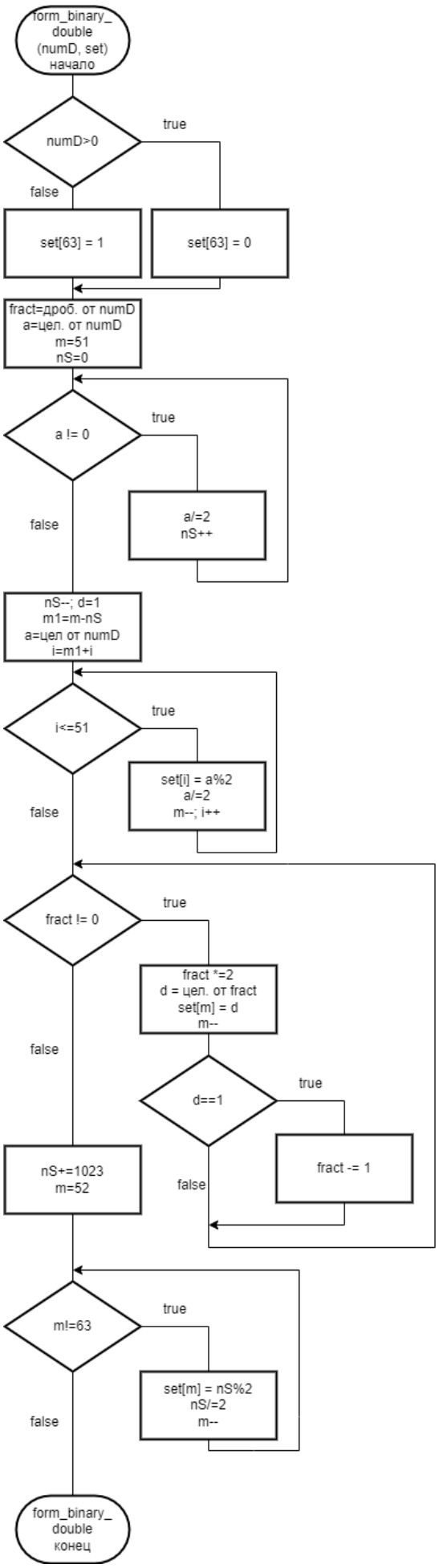
Мельник Д. А. – разработка алгоритмов, написание отчёта

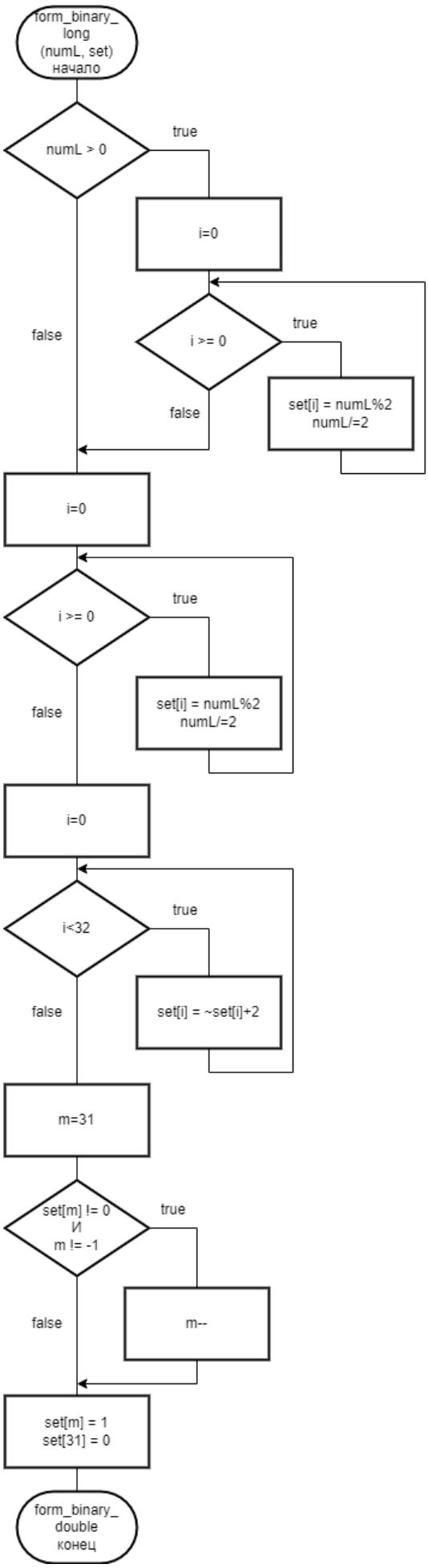
Лепов А. В. – написание и отладка кода программы

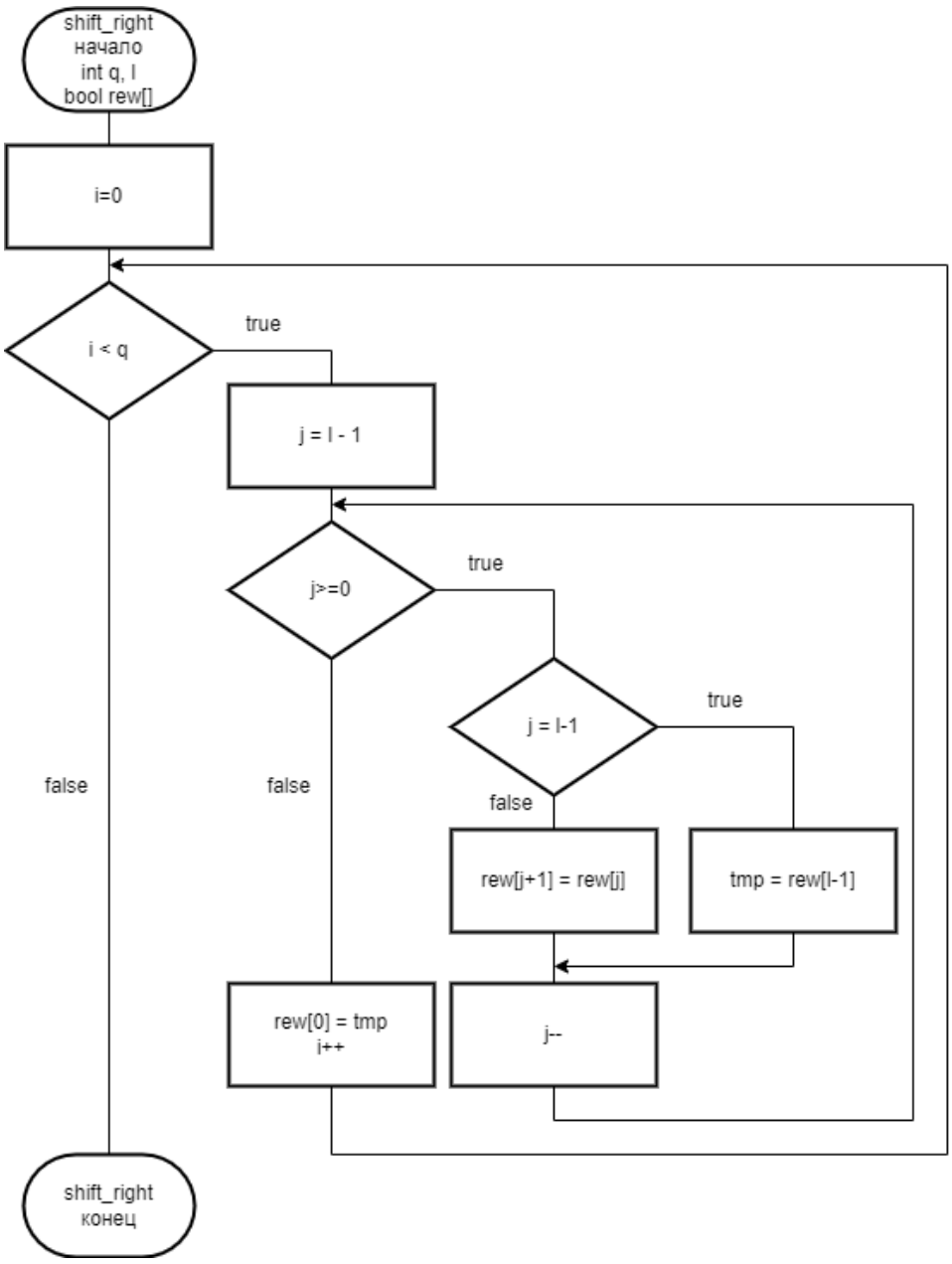
БЛОК-СХЕМА АЛГОРИТМА

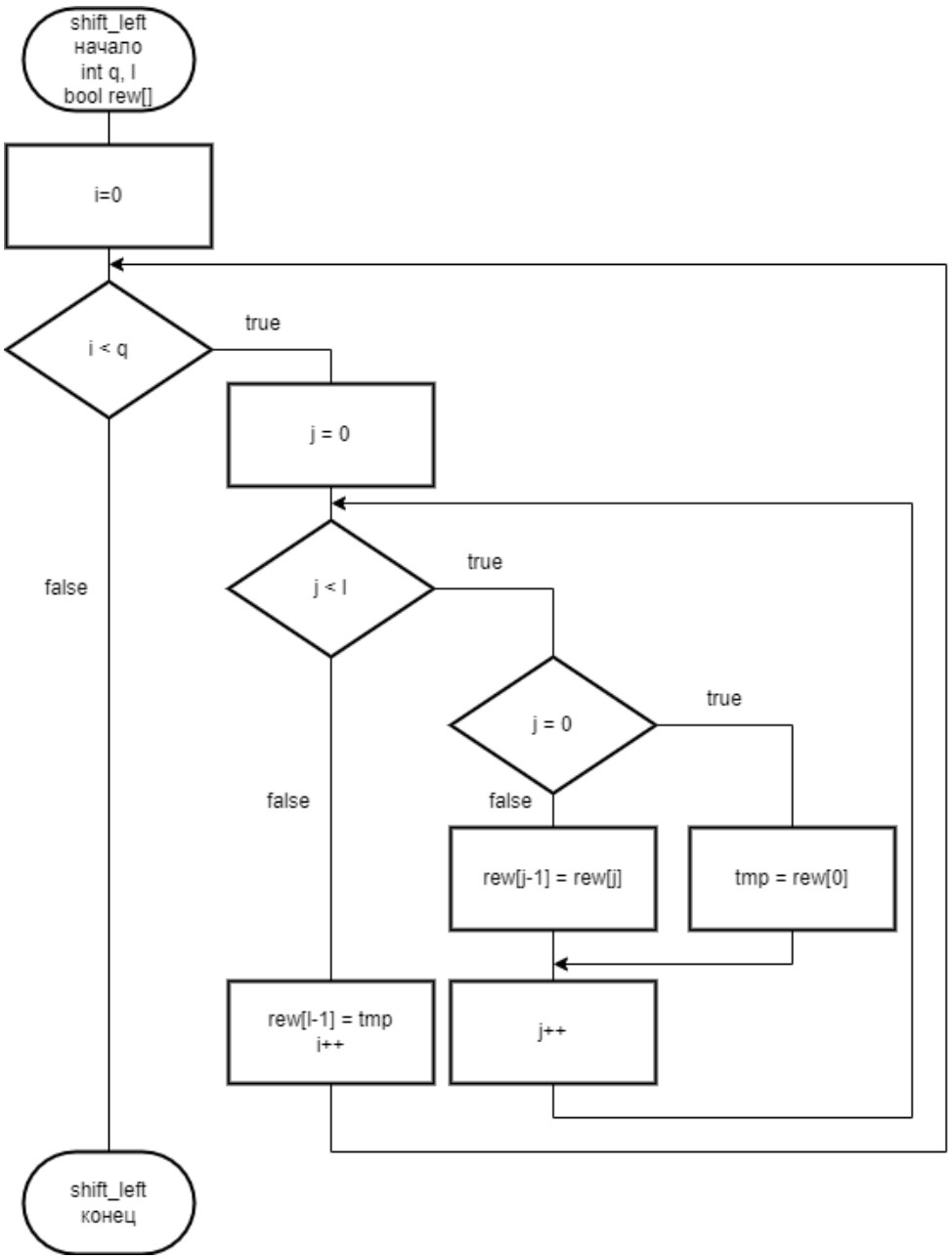


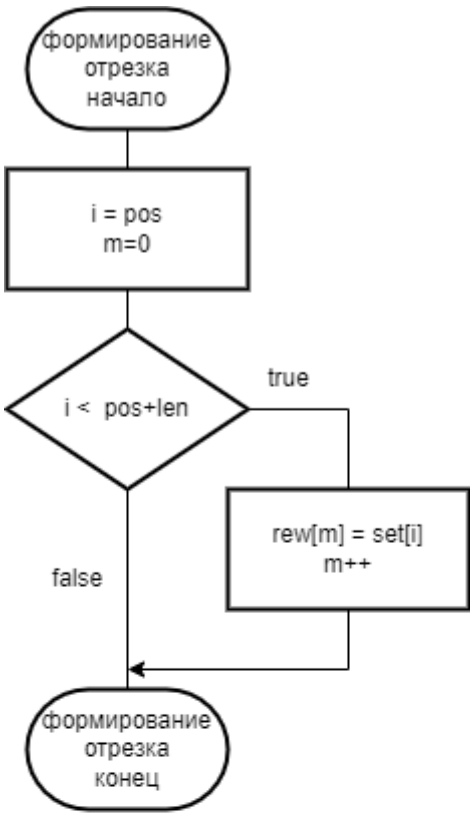

back to dec
double
начало
bool set, sign
sumD=0
m=0
S=0
i=52
i <= 62
true
S+=set[i] * pow(2,m)
m++
i++
false
S-=1023
m=51
i=1
i<=S
true
sumD+=set[m] *
pow(2,S-i)
m--
i++
false
sumD+= pow(2,S)
d=-1
i=m
i>=0
true
sumD+=set[i] *
pow(2,d)
d--
i--
false
sign = false
true
sumD *= -1
false
back to dec
double
конец

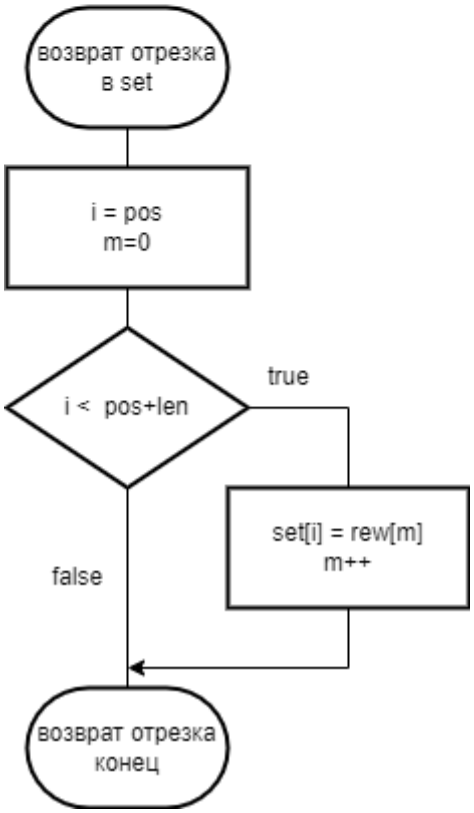

form_binary_
double
(numD, set)
начало
numD>0
true
false
set[63] = 1
set[63] = 0
fract=дроб. от numD
a=цел. от numD
m=51
nS=0
a != 0
true
false
a/=2
nS++
nS--; d=1
m1=m-nS
a=цел от numD
i=m1+i
i<=51
true
false
set[i] = a%2
a/=2
m--; i++
fract != 0
true
false
fract *=2
d = цел. от fract
set[m] = d
m--
d==1
true
false
fract -= 1
nS+=1023
m=52
m!=63
true
false
set[m] = nS%2
nS/=2
m--
form_binary_
double
конец

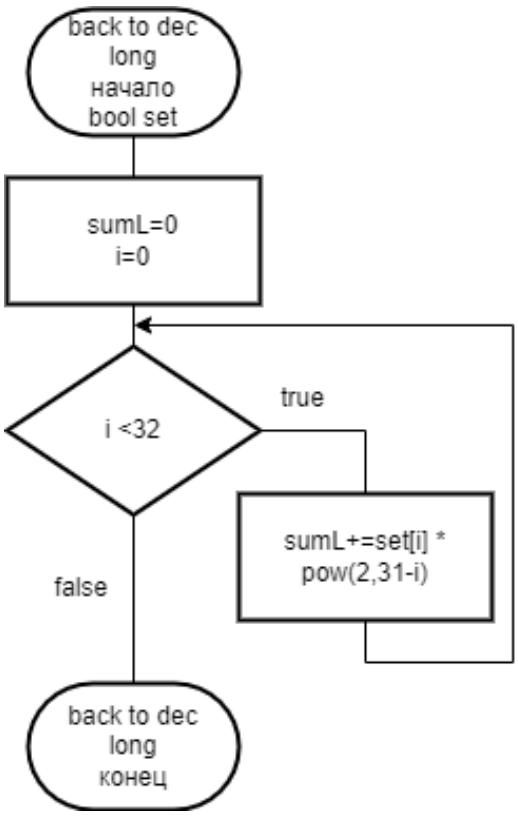

form_binary_
long
(numL, set)
начало
numL > 0
true
false
i=0
i >= 0
true
false
set[i] = numL%2
numL/=2
i=0
i >= 0
true
false
set[i] = numL%2
numL/=2
i=0
i<32
true
false
set[i] = ~set[i]+2
m=31
set[m] != 0
И
m != -1
true
false
m--
set[m] = 1
set[31] = 0
form_binary_
double
конец


shift_right
начало
int q, l
bool rew[]
i=0
i < q
true
false
j = l - 1
j>=0
true
false
j = l-1
true
false
rew[j+1] = rew[j]
tmp = rew[l-1]
rew[0] = tmp
i++
j--
shift_right
конец


shift_left
начало
int q, l
bool rew[]
i=0
i < q
true
false
j = 0
j < l
true
false
j = 0
true
false
rew[j-1] = rew[j]
tmp = rew[0]
rew[l-1] = tmp
i++
j++
shift_left
конец


формирование отрезка начало
i = pos
m=0
i < pos+len
true
rew[m] = set[i]
m++
false
формирование отрезка конец


возврат отрезка в set
i = pos
m=0
i < pos+len
true
set[i] = rew[m]
m++
false
возврат отрезка конец


back to dec long
начало
bool set
sumL=0
i=0
i <32
true
sumL+=set[i] *
pow(2,31-i)
false
back to dec long
конец

СЛОВЕСНОЕ ОПИСАНИЕ АЛГОРИТМА (ПОЯСНЕНИЯ)

main()

Для определения типа входной переменной, а также для проверки корректности ввода чисел (отсутствие лишних арифметических знаков и букв) , число вводится строкой.

Строка проходит проверку на наличие букв, лишних точек и минусов. Пока все проверки не будут пройдены, пользователь будет иметь попытку ввести число заново (при ошибках ему выводится соответствующее сообщение). Проверка на корректность ввода на схеме отсутствуют, так как не связана с представлением чисел в памяти ЭВМ.

По наличию единственной точки в записи числа определяется его тип (по ТЗ либо long (1), либо double (0))

В зависимости от типа данных, переменной соответствующего типа присваивается значение в строке (через atoi (long) или atof (double)).

В соответствии с типом данных, long или double, создаётся массив set элементов типа bool размером 32 и 64 соответственно.

С помощью функций form_binary_double и form_binary_long в массив set записывается двоичное представление чисел double или long соответственно.

Далее вводятся три параметра: первая позиция изменяемого отрезка, длина этого отрезка, количество битов сдвига. На каждый из параметров наложены ограничения: позиция может быть выбрана в long любая кроме позиции знака, в double можно выбрать только внутри мантиссы. Длина ограничена расстоянием от позиции до конца кода представления, количество битов сдвига не ограничено.

Далее формируется отрезок rew массива битов set, который подлежит сдвигу.

Пользователю даётся выбор стороны сдвига (защита от ввода бездумного символа присутствует). После выбора стороны сдвига при помощи

функций shift_left и shift_right отрезок rew массива set смещается влево или вправо соответственно на указанное количество битов сдвига.

Далее выполняется возвращение элементов (битов) массива rew на прежние места (по индексам) в set.

На основании результирующего массива битов set производится перевод назад в десятичную систему в соответствии с типом через функции back_to_decimal_long(/double).

На выход поступает число после сдвига.

Алгоритмы перевода чисел long и double в двоичное представление ЭВМ.

В работе учтено, что целые положительные числа хранятся в ЭВМ прямым кодом, отрицательные – дополнительным, следовательно, использованы алгоритмы перевод десятичного числа в двоичное, получение обратного кода через инверсию прямого с прибавлением единицы. Симметричный алгоритм использован для обратного преобразования.

Для представления числа с плавающей точкой была использована функция библиотеки <cmath> modf. С её помощью вычислялась целая и дробная части числа с плавающей точкой. Далее, целая часть переводится в соответствующий участок мантиссы в сетке [51–51-P], при этом вычисляется порядок. После, дробная часть переводится в двоичный вид и записывается в мантиссу за целой [51-P—0]. Мнимая единица целой части согласно алгоритму в мантиссу не попадает. К вычисленному порядку прибавляется 1023, порядок записывается в двоичном виде в соответствующую часть разрядной сетки 62—52]. Обратный порядок перевода (функции back_to_decimal_long(/double)) симметричны функциям перевода из десятичного вида в представление ЭВМ (form_binary_long(/double)).

Функции сдвига (shift_left(/right))

Учтена цикличность сдвига, соответственно каждый последний бит должен переходить в начало сетки при сдвиге вправо и каждый первый бит должен переходить сразу в конец при сдвиге влево. Алгоритм путём перевбора

двгиет элементы либо влево, либо вправо, при этом пограничные элементы сохраняются во временную переменную tmp и записываются после одной проходки перебора. Количество проходок перебора = количество битов сдвига.

На случай, если пользователь случайно введёт где-нибудь в программе символ кириллицы, предусмотрено ручная установка кодировок в ОП Windows. В консоли надо иметь соответсвующий шрифт **lucida console** для корректной работы программы. **Ввод дробных значений осуществляется с запятой, а не с точкой.** Данная информация доводится до пользователя.

Осуществлена защита от букв, нескольких применений запятых и минусов, абсурдно больших чисел, выходящих за пределы вместимости типов данных.

ТЕКСТ ПРОГРАММЫ

```
#include<iostream>
#include<string.h>
#include<cmath>
#include<windows.h>

using namespace std;

int find_simbol(string str1, char str2[]){
    int d = 0;
    for (int i =0; i<str1.size();i++){
        for (int j=0; j<strlen(str2); j++){
            if (str1[i]==str2[j]) d++;
        }
    }
    return d;
}

void form_binary_long (long numL, bool set[]){
    bool sign;
    numL >= 0 ? sign = true : sign = false;
    switch (sign)
    {
    case true:
        for (int i = 31; i>=0; i--){
            set[i]=numL%2;
            numL/=2;
        }
        break;
    case false:
        for (int i = 31; i>=0; i--){
            set[i]=numL%2;
            numL/=2;
        }
        for (int i=0;i<32;i++){
```

```
            set[i]=~set[i]+2;
        }
        int m = 31;
        while ((set[m]!=0)&&(m!=-1)){
            m--;
        }
        if (set[31]==1) set[31]=0;
        set[m] = 1;
        break;
    }
    cout << "binary form of a number: \n";
    cout << "|0------------------------------31|\n";
    cout << "|";
    for (int i=0; i<32; i++) cout << set[i];
    cout << "  |";
}

void form_binary_double (double numD, bool set[]){
    bool sign;
    double drob, fract;
    int nS, d, m1;
    numD >= 0 ? sign = true : sign = false;
    sign == true ? set[63] = 0 : set[63] = 1;
    if (sign == false) numD*=-1;

    fract = modf(numD, &drob);
    int m=51;
    int a = static_cast<int>(drob);
    nS=0;
    while (a!=0){
        a/=2;
        nS++;
    }
    nS--;
    d=1;
    m1=m-nS;
```

```
    a = static_cast<int>(drob);
    for (int i = m1+1; i<=51; i++) {
        set[i]=a%2;
        a/=2;
        m--;
    }
    while (fract!=0){
        fract*=2;
        d=static_cast<int>(fract);
        set[m] = d;
        if (d==1) fract-=1;
        m--;
    }
    nS+=1023;
    m=52;
    while (m!=63){
        set[m] = nS%2;
        nS/=2;
        m++;
    }
    cout << "binary form of a number (S-P-M): \n";
    cout << "63|62" << "         " << "52|51" << "             "<<"
0|\n";
    cout <<set[63] << " |  ";
    for (int i =62; i>=52; i--) cout << set[i];
    cout << "  |  ";
    for (int i =51; i>=0; i--) cout << set[i];
    cout<<"|";
}

void shift_right (bool rew[], int quantity, int length){
    bool tmp=0;
    for(int i=0; i<quantity; i++)
        {
            for(int j=length-1; j>=0; j--)
            {
```

```
                if (j==length-1)
                    tmp=rew[length-1];
                else
                    rew[j+1]=rew[j];
            }
            rew[0]=tmp;
        }
}
void shift_left (bool rew[], int quantity, int length){
    bool tmp = 0;
    for(int i=0; i<quantity; i++)
        {
            for(int j=0; j<length; j++)
            {
                if (j==0)
                    tmp=rew[0];
                else
                    rew[j-1]=rew[j];
            }
            rew[length-1]=tmp;
        }
}

double back_to_decimal_double(bool set[], bool sign){
    double sumD=0;
    int m, S, d;
    m=0;
    S=0;
    for (int i = 52; i<=62; i++){
        S+=set[i]*pow(2,m);
        m++;
    }
    S-=1023;
    m=51;
    cout << endl;
    for (int i = 1; i<=S;i++){
```

```
        sumD+=set[m]*pow(2,S-i);
        m--;
    }
    sumD = sumD + pow(2,S);
    d=-1;
    for (int i=m;i>=0;i--) {
        sumD = sumD + set[i]*(pow(2,d));
            d--;
    }
    if (set[63]==1) sumD= sumD*(-1);
    return sumD;
}

long back_to_decimal_long (bool set[]){
    long sumL=0;
    for (int i=0; i<32; i++) sumL+=set[i]*pow(2,31-i);
    return sumL;
}

int int_vvod(){
    int pos=0, d;
    bool tess, us;
    char str1[100], simbols[]="0123456789-";
    SetConsoleCP(1251);
     SetConsoleOutputCP(1251);
    tess=true;
    while (tess){
    cin >> str1;
    us = true;
    d=0;
    for (int i=0; i<strlen(str1);i++){
        for (int j=0; j<strlen(simbols);j++){
            if (str1[i]==simbols[j]) d++;
        }
    }
    d==strlen(str1) ? us = true : us = false;
```

```
    if (us){
        if                                          (((str1[0]!='-
')&&(strlen(str1)<strlen("2147483647")))||((str1[0]=='-
')&&(strlen(str1)<strlen("-2147483647"))))  {pos  =  atoi(str1);
tess=false;}
            if
((strlen(str1)==strlen("2147483647"))||(strlen(str1)==strlen("-
2147483647"))){
                if
((strcmp(str1,"2147483647")<=0)||(strcmp(str1,"-
2147483647")<=0)) {
                    pos = atoi(str1);
                    tess=false;
                }
            }
            if  (tess!=false)cout  <<  "\nnum  is  too  large  for
int\nenter new one [-2147483648...2147483647]: ";
        }
        else cout << "please, use numbers and mines\nenter new
one: ";
    }
    return pos;

}

int main(){
    long numL, sumL = 0;
    double numD, sumD=0.0;
    bool sign, *set, tes = true, tes1 = false, tes2=true, *rew;
    int typ, position, quantity, length, tm, tt;
    int d, m, ccorddot=0;
    char side, str[100];
    char  M[]  =  "0123456789,-",  L1[]  =  "-2147483648",  L2[]  =
"2147483647",   D1[]   =   "-9223372036854775808,0",   D2[]   =
"9223372036854775807,0";
    cout << "Attention! to enter doble, use a comma, not a dot.\n";
```

```
    cout << "enter your number: ";
    while (tes){
        cin >> str;
        tes2=true;
        d=0;
        tt=0;
        tm=0;
        d=find_simbol(str, M);
        for (int i=0; i<strlen(str);i++){
            if (str[i]==',') {tt++; ccorddot = i;}
            if (str[i]=='-') tm++;
        }

        if((d==strlen(str))&&(tes2)&&(tm<=1)&&(tt<=1)){
            tes = false;

         }
         if ((tt==0)&&(tes==false)){
            if             (((str[0]             ==             '-
')&&(strlen(str)>strlen(L1)))||((str[0]!='-
')&&(strlen(str)>strlen(L2)))){
                tes = true;
            }
            if             (((str[0]             ==             '-
')&&(strlen(str)==strlen(L1)))||((str[0]!='-
')&&(strlen(str)==strlen(L2)))){
                if(str[0]=='-'){
                    if(strcmp(str,L1)>0) tes = true;
                }
                if(str[0]!='-'){
                    if(strcmp(str,L2)>0) tes = true;
                }
            }
            if (tes==true) {
                cout << "if you entered a number that is too large
or too small:\n";
```

```
                cout << "\nFor data type long, acceptable values
are [-2147483648...2147483647], for data type double - [-
9223372036854775808,0...9223372036854775807,0].\n\n";
                cout<< "this number is too large for long
type\nenter new one: ";
            }
        }
        if ((tt==1)&&(tes==false)){
            if (((str[0] == '-')&&((ccorddot)>20))||((str[0]!='-
')&&((ccorddot)>19))){
                tes = true;
            }
            if (((str[0] == '-')&&((ccorddot)==20))||((str[0]!='-
')&&((ccorddot)==19))){
                if(str[0]=='-'){
                    if(strcmp(str,D1)>0) tes = true;
                }
                if(str[0]!='-'){
                    if(strcmp(str,D2)>0) tes = true;
                }
            }
            if (tes==true) {
                cout << "if you entered a number that is too large
or too small:\n";
                cout << "\nFor data type long, acceptable values
are [-2147483648...2147483647], for data type double - [-
9223372036854775808,0...9223372036854775807,0].\n\n";
                cout<< "this number is too large for double
type\nenter new one: ";
            }
        }
        if (str[0]=='.') {tes=true; cout << "incorrect using a
comma\n";}
        if (d!=strlen(str)) cout << "\nyou can only use numbers ,
and -\nenter new one: ";
```

```
        if (tes2==false) cout << "\nuse , not a . \nenter new
one: ";
        if (tm>1) cout << "\nif you want to use a mines, then use
it once.\nenter new one: ";
        if (tt>1) cout << "\nif you want to use a comma, then use
it once.\nenter new one: ";

    }
    tt == 1 ? typ = 0 : typ = 1;

    //Enter num
    /*----------------------------------------*/
    switch (typ)
    {
    case 0:{
        cout << "you are using double\n" << endl;
        numD=atof(str);

        set = new bool [64];
        for (int i = 0; i<64; i++) set[i] = 0;
        form_binary_double(numD,set);
        break;
    }
//long
    case 1:
        cout<<"you are using long type\n";
        numL=atoi(str);
        cout << "numL = " << numL << endl;
        set = new bool[32];
        for (int i=0; i<32; i++) set[i] = 0;
        form_binary_long(numL,set);
    }

    cout << endl;
    switch (typ)
    {
```

```
    case 0: //double
        cout << "Enter the position: "; //position
        tes = true;
        while (tes){
            position = int_vvod();
            switch (position)
            {
            case 0 ... 51:
                cout << "the position is entered correctly" <<
endl;
                tes=false;
                break;
            default:
                cout << "the value should be in the range from 0
to 51, enter a new one: ";
            }
        }
        cout << "Enter the length: ";
        tes = true;
        while (tes){//length
            length = int_vvod();
            if (length > 51-position){
                cout << "the value should be in the range from 1
to "<< 51-position << ", enter a new one: ";
            }
            else tes = false;
        }
        cout << "Enter the quantity: ";
        quantity = int_vvod();
        while (quantity<0)
        {
            cout << "quantiy must be >= 0\nenter new one: ";
            quantity = int_vvod();

        }
        break;
```

```
    case 1: //long
        cout << "Enter the position: "; //position
        tes = true;
        while (tes){
            position = int_vvod();
            switch (position)
            {
            case 1 ... 31:
                cout << "the position is entered correctly" <<
endl;
                tes=false;
                break;
            default:
                cout << "the value should be in the range from 1
to 31, enter a new one: ";
            }
        }
        int b = 31-position;
        cout << "Enter the length: ";
        tes = true;
        while (tes){//length
            length = int_vvod();
            if (length > 31-position+1){
                cout << "the value should be in the range from 1
to "<< 31-position+1 << ", enter a new one: ";
            }
            else tes = false;
        }
        cout << "Enter the quantity: ";
        quantity = int_vvod();
        while (quantity<0)
        {
            cout << "quantiy must be >= 0\nenter new one: ";
            quantity = int_vvod();
        }
        break;
```

```
}
rew = new bool [length];
for (int i =0; i<length; i++) rew[i]=0;
m=0;

for (int i = position; i< position + length; i++){
    rew[m] = set[i];
    m++;
}
cout << "Enter the side (r = ->; l = <-): ";
cin >> side;
while ((side!='r')&&(side!='l')){
    cout << "enter r or l: ";
    cin >> side;
}
cout << endl;

cout << "the segment before the shift\n";
for (int i = 0; i<length; i++) cout << rew[i] << " ";
cout << endl;

switch (side)
{
case 'r':
    shift_right(rew,quantity,length);
    break;

case 'l':
    shift_left(rew,quantity,length);
    break;
}

cout << "the segment after the shift\n";
for (int i = 0; i<length; i++) cout << rew[i] << " ";
cout << endl;
```

```
    m=0;
    for (int i=position; i<position+length;i++){
        set[i]=rew[m];
        m++;
    }
    cout << endl;
    switch (typ)
    {
    case 0:
        sumD=back_to_decimal_double(set,sign);
        cout << "numD = " << sumD << endl;
        break;
    case 1:
        sumL=back_to_decimal_long(set);
        cout << "binary form of a number: \n";
        cout << "|0-------------------------------31|\n";
        cout << "|";
        for (int i=0; i<32; i++) cout << set[i];
        cout << "  |";
        if (sumL<0) sumL+=1;
        cout << "\nnumL = " << sumL;
    }
    delete [] set;
    delete [] rew;
    return 0;
}
```

ТЕСТОВЫЕ ПРИМЕРЫ

В программе выполнены промежуточный выводы для ручной проверки работы ЭВМ.

Целые числа (long):

```
enter your number: 123
you are using long type
numL = 123
binary form of a number:
|0-------------------------------31|
|00000000000000000000000001111011  |
Enter the position: 25
the position is entered correctly
Enter the length: 5
Enter the quantity: 2
Enter the side (r = ->; l = <-): r

the segment before the shift
1 1 1 1 0
the segment after the shift
1 0 1 1 1

binary form of a number:
|0-------------------------------31|
|00000000000000000000000001011111  |
numL = 95
Process returned 0 (0x0)   execution time : 24.188 s
Press any key to continue.
```

```
enter your number: -123
you are using long type
numL = -123
binary form of a number:
|0-------------------------------31|
|11111111111111111111111110000101  |
Enter the position: 25
the position is entered correctly
Enter the length: 5
Enter the quantity: 2
Enter the side (r = ->; l = <-): l

the segment before the shift
0 0 0 0 1
the segment after the shift
0 0 1 0 0

binary form of a number:
|0-------------------------------31|
|11111111111111111111111110010001  |
numL = -110
Process returned 0 (0x0)   execution time : 18.190 s
Press any key to continue.
```

```
enter your number: 99999999999999999
if you entered a number that is too large or too small:

For data type long, acceptable values are [-2147483648...2147483647], for data

this number is too large for long type
enter new one: ф

you can only use numbers . and -
enter new one: 99999999999999999999999999
if you entered a number that is too large or too small:

For data type long, acceptable values are [-2147483648...2147483647], for data

this number is too large for long type
enter new one: s

you can only use numbers . and -
enter new one: 56
you are using long type
numL = 56
binary form of a number:
|0-------------------------------31|
|00000000000000000000000000111000  |
Enter the position: 9999999999999999999

num is too large for int
enter new one [-2147483648...2147483647]: ы
please, use numbers and mines
enter new one: 56
the value should be in the range from 1 to 31, enter a new one: 5
the position is entered correctly
Enter the length: 8989865461349816546165

num is too large for int
enter new one [-2147483648...2147483647]: s
please, use numbers and mines
enter new one: 49877987446514968

num is too large for int
enter new one [-2147483648...2147483647]: 5
Enter the quantity: ы
please, use numbers and mines
enter new one: 56
Enter the side (r = ->; l = <-): л
enter r or l: п
enter r or l: po
enter r or l: enter r or l: r

the segment before the shift
0 0 0 0 0
the segment after the shift
0 0 0 0 0

binary form of a number:
|0-------------------------------31|
|00000000000000000000000000111000  |
numL = 56
Process returned 0 (0x0)   execution time : 77.900 s
Press any key to continue.
```

```
enter your number: 0
you are using long type
numL = 0
binary form of a number:
|0-------------------------------31|
|00000000000000000000000000000000  |
Enter the position: 5
the position is entered correctly
Enter the length: 5
Enter the quantity: 5
Enter the side (r = ->; l = <-): r

the segment before the shift
0 0 0 0 0
the segment after the shift
0 0 0 0 0

binary form of a number:
|0-------------------------------31|
|00000000000000000000000000000000  |
numL = 0
Process returned 0 (0x0)   execution time : 15.014 s
Press any key to continue.
```

```
enter your number: -89
you are using long type
numL = -89
binary form of a number:
|0-------------------------------31|
|11111111111111111111111110100111  |
Enter the position: 25
the position is entered correctly
Enter the length: 5
Enter the quantity: 3
Enter the side (r = ->; l = <-): l

the segment before the shift
0 1 0 0 1
the segment after the shift
0 1 0 1 0

binary form of a number:
|0-------------------------------31|
|11111111111111111111111110101011  |
numL = -84
Process returned 0 (0x0)   execution time : 28.620 s
Press any key to continue.
```

Числа с плавающей точкой.

```
enter your number: -15.375

you can only use numbers , and -
enter new one: -15,375
you are using double

binary form of a number (S-P-M):
63|62          52|51                                                          0|
1 |  10000000010  |  1110110000000000000000000000000000000000000000000000|
Enter the position: 45
the position is entered correctly
Enter the length: 5
Enter the quantity: 2
Enter the side (r = ->; l = <-): r

the segment before the shift
0 1 1 0 1
the segment after the shift
0 1 0 1 1


numD = -15.625

Process returned 0 (0x0)   execution time : 80.474 s
Press any key to continue.
```

```
enter your number: 15,375
you are using double

binary form of a number (S-P-M):
63|62          52|51                                                          0|
0 |  10000000010  |  1110110000000000000000000000000000000000000000000000|
Enter the position: 45
the position is entered correctly
Enter the length: 5
Enter the quantity: 2
Enter the side (r = ->; l = <-): r

the segment before the shift
0 1 1 0 1
the segment after the shift
0 1 0 1 1


numD = 15.625

Process returned 0 (0x0)   execution time : 22.969 s
Press any key to continue.
```

```
enter your number: 125.625

you can only use numbers , and -
enter new one: 125,625
you are using double

binary form of a number (S-P-M):
63|62           52|51                                                          0|
0 |  10000000101  |  1111011010000000000000000000000000000000000000000000|
Enter the position: 40
the position is entered correctly
Enter the length: 10
Enter the quantity: 5
Enter the side (r = ->; l = <-): l

the segment before the shift
0 0 0 1 0 1 1 0 1 1
the segment after the shift
1 1 0 1 1 0 0 0 1 0


numD = 116.422

Process returned 0 (0x0)   execution time : 31.486 s
Press any key to continue.
```

```
enter your number: a

you can only use numbers , and -
enter new one: ы

you can only use numbers , and -
enter new one: --152

if you want to use a mines, then use it once.
enter new one: -125..36

you can only use numbers , and -
enter new one: -125,,58

if you want to use a common, then use it once.
enter new one: 115,25
you are using double

binary form of a number (S-P-M):
63|62           52|51                                                          0|
0 |  10000000101  |  1100110100000000000000000000000000000000000000000000|
Enter the position: 555
the value should be in the range from 0 to 51, enter a new one: 45
the position is entered correctly
Enter the length: 8
the value should be in the range from 1 to 6, enter a new one: 6
Enter the quantity: aa
please, use numbers and mines
enter new one: -1
quantiy must be >= 0
enter new one: 56
Enter the side (r = ->; l = <-): l

the segment before the shift
0 1 1 0 0 1
the segment after the shift
1 0 0 1 0 1


numD = 116.75

Process returned 0 (0x0)   execution time : 76.672 s
Press any key to continue.
```

СРЕДА РАЗРАБОТКИ

Программа выполнена и откомпелирована в среде разработки **Code Blocks**.

В консоли необходимо установить шрифт **lucida console,** иначе, при проверки ввода данных программа не будет учитывать символы кириллицы, который пользователь может по неосторожности ввести.