**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**кафедра ВТ**

**ОТЧЕТ**

**по лабораторной работе по дисциплине**

**«Организация ЭВМ и систем»**

Студент группы 8374 Пихтовников К.С.

Преподаватель: Павлов С.М.

Санкт-Петербург

2019

**Цель работы:**

Исследовать внутреннее представление чисел различных типов данных. Разобраться со структурой внутреннего представления компьютера. Научиться работать с цветом и масштабированным графиком.

**ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ:**

1. Разработать программу, которая принимает число следующих типов данных: bool,char,short int,unsigned short int, int, unsigned int, float, double в любой системе счисления (от 2 до 128-ой). Программа должна показать на экране внутреннее представление числа в двоичной системе счисления.

2. В соответствии с номером индивидуального задания (25) разработать программу, которая меняет местами группы бит, заданные пользователем.

3. Разработать программу, которая меняет цвета групп бит, которые были поменяны местами во 2 задании. Цвета бит выбирается в правом верхнем углу. Нажимая клавиши enter и стрелки, можно выбрать нужный цвет. Также написать динамический вывод двоичного представления числа.

4. Разработать программу, которая будет выводить график фазовой модуляции, отражающий внутреннее представление числа в двоичной СС. Дать возможность пользователю изменять масштаб графика.

**Задание №1**

Ниже представлены основные блоки программы, с помощью которых мы получаем на экране внутреннее представление чисел разных типов данных в двоичной системе счисления.

template <class T>

union types {

T d;

char c[sizeof(T)];

};

template<class T>

void output\_bynary\_system(T number) {

bool bit;

int k = sizeof(T) - 1, i = 0;

types<T> u;

u.d = number;

bool ch[8 \* sizeof(T)];

while (k >= 0)

{

i = 7;

while (i >= 0)

{

bit = u.c[k] & (1U << i);

cout << bit;

//i--;

\_asm

{

mov eax,i

dec i

}

}

//k--;

\_asm

{

mov eax,k

dec k

}

}

}

**Краткое описание:**

В программе есть 8 функций для каждого соответствующего типа данных. Вот, например, 3 из них.

void short\_int(int number\_system,int count)

{

short int number = 0;

number = entering\_number(number, number\_system, count);

dynamic\_output(number);

number = input\_group\_bit(number, number\_system);

chart(number);

}

void \_int(int number\_system,int count)

{

int number = 0;

number = entering\_number(number, number\_system, count);

dynamic\_output(number);

number = input\_group\_bit(number, number\_system);

chart(number);

}

void \_double(int number\_system,int count)

{

double number = 0;

number = entering\_number(number, number\_system, count);

dynamic\_output(number);

number = input\_group\_bit(number, number\_system);

chart(number);

}

Внутри тела одной из 8 функции, это зависит от того с каким типом данных работает пользователь , в функции

template<class T>

T entering\_number(T number,int number\_system,int count)

вводится число. Затем это функция возвращает значение этого числа и присваивает переменной number.В свою очередь же это значение посылается в функцию output\_bynary\_system(T number).

Рассмотрим подробнее как работает output\_bynary\_system(T number):

Переменной types<T> u присваивается значение T number.Затем в цикле выполняется bit = u.c[k] & (1U << i).Это в переменную типа bool присваивается значение бита, стоящего на i-ом месте в числе number.Следующая строка демонстрирует вывод этого бита на экран:

cout << bit;

**Задание №2**

**Краткое описание:**

В качестве примера, возьмем эту функцию.

void \_int(int number\_system,int count)

{

int number = 0;

number = entering\_number(number, number\_system, count);

dynamic\_output(number);

number = input\_group\_bit(number, number\_system);

chart(number);

}

template<class T>

T input\_group\_bit(T number, int number\_system) –функция, у которых есть 2 параметра.1-число,которое ввел пользователь,2-система счисления, тоже введенная пользователем.

Внутри этой функции есть переменные deque<int> s1 , s2, в которые записываются номера битов, который выбрал пользователь. Затем эти переменные и значение числа number посылаются в функцию T swap\_bites(…).

Подробнее рассмотрим работу функции T swap\_bites(…):

В массив array\_bites\_number[8\*sizeof(T)] в цикле записывается битовое представление числа number.

int size\_number\_byte = sizeof(T) - 1, array\_bites\_number[8\*sizeof(T)];

types<T> u;

u.d = number;

count = 8\*sizeof(T) - 1;

while (size\_number\_byte >= 0)

{

i = 7;

while (i >= 0)

{

array\_bites\_number[count]= u.c[size\_number\_byte] & (1U << i);;

count--;

i--;

}

size\_number\_byte--;

}

Затем идет проверка на размер переменных s1 и s2. Т.к пользователь мог ввести одинаковое кол-во битов как в 1 группе,так и во 2 или же разное.Далее происходит перестановка группы бит в массиве

array\_bites\_number. Здесь приведен фрагмент программы,если бы пользователь ввел одинаковое кол-во битов в обеих группах.

if (s1.size() == s2.size())

{

k1 = s1.front();

k2 = s2.front();

while (k1 <= s1.back())

{

bit = array\_bites\_number[k2];

array\_bites\_number[k2] = array\_bites\_number[k1];

array\_bites\_number[k1] = bit;

k1++;

k2++;

}

}

Затем этот массив и значение number посылаются в функцию

T assigning\_bits\_to\_an\_array(..).

Далее в теле цикла мы сравниваем каждый бит массива array\_bites\_number и каждый бит числа number.И если эти биты не равны, инвертируем соответствующий бит числа number.В конце функция возвращает значение переменной number.

T assigning\_bits\_to\_an\_array(T number,int array\_bites\_number[8\*sizeof(T)])

{

types<T>u;

u.d = number;

int k = sizeof(T) - 1,i=0,count=8\*sizeof(T)-1;

bool bit1,bit2,bit;

int x = 0;

while (k >= 0)

{

i = 7;

while (i >= 0)

{

bit1 = array\_bites\_number[count];

bit2 = u.c[k] & (1U << i);

if (bit1 != bit2)

{

u.c[k] = u.c[k] ^ ((1U << i));

}

--i;

count--;

x++;

}

k--;

}

number = u.d;

return number;

}

**Задание №3**

**Текст программы и краткое описание:**

Ниже представлена часть программы, требуемая для изменения цвета битов.

Функция **SetColor()** принимает два параметра. Первый параметр отвечает за цвет теста в консоли, второй за цвет фона(консоли).

void SetColor(int text, ConsoleColor background)

{

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hStdOut, (WORD)((background << 0) | text));

}

Описание функции **getColor()** :

Всего реализовано 12 цветов. В переменную целого типа key записывается код нажатой клавиши пользователем. Это может клавиша-стрелка вверх/вниз и enter.На другие клавиши программа реагировать не будет. Если нажаты клавиши-стрелки, то в зависимости от того какая клавиша нажата выполняется определенное условие и увеличивается/уменьшается переменная целого типа y.

Если была нажата клавиша enter, то программа заходит в оператор switch(y)-case и в зависимости от y,функция getColor возвращает текущий цвет рисования

ConsoleColor getColor()

{

int key = 0;

int x = 30, y = 3;

gotoxy(0, 1);

cout << endl << "Выберите цвет от 1 до 12";

SetColor(Blue, Black);

cout <<endl<< "1.Blue" << endl;

SetColor(Green, Black);

cout << "2.Green" << endl;

SetColor(Cyan, Black);

cout << "3.Cyan" << endl;

SetColor(Red, Black);

cout << "4.Red" << endl;

SetColor(Magenta, Black);

cout << "5.Magenta" << endl;

SetColor(LightGreen, Black);

cout << "6.LightGreen" << endl;

SetColor(LightMagenta, Black);

cout << "7.LightMagenta" << endl;

SetColor(Yellow, Black);

cout << "8.Yellow" << endl;

SetColor(LightBlue, Black);

cout << "9.LightBlue" << endl;

SetColor(LightCyan, Black);

cout << "10.LightCyan" << endl;

SetColor(LightRed, Black);

cout << "11.LightRed" << endl;

SetColor(Brown, Black);

cout << "12.Brown" << endl;

SetColor(White, Black);

gotoxy(30,3);

cout << "<==";

while (true) {

if (key == down && y < 14) {

clear\_screen(30, 33, y, y);

y++;

gotoxy(x, y);

cout << "<==";

}

if (key == up && y>3) {

clear\_screen(30, 33, y, y);

y--;

gotoxy(x, y);

cout << "<==";

}

if (key == \_enter)

{

switch (y) {

case 3:

return Blue;

case 4:

return Green;

case 5:

return Cyan;

case 6:

return Red;

case 7:

return Magenta;

case 8:

return LightGreen;

case 9:

return LightMagenta;

case 10:

return Yellow;

case 11:

return LightBlue;

case 12:

return LightCyan;

case 13:

return LightRed;

case 14:

return Brown;

}

}

key = \_getch();

}

}

**Функция “clear\_screen” очищает часть экрана**

void clear\_screen(int x1, int x2, int y1, int y2) {

for (int i = y1; i <= y2; ++i)

for (int j = x1; j <= x2; ++j)

{

gotoxy(j, i);

cout << ' ';

}

}

gotoxy(x,y)- помещает курсор экрана в точку с координатами х,у.

**Функция динамического вывода двоичного представления числа:**

Фунция принимает значение числа number,координату экрана x и y.

Вывод происходит так: если бит стоит на четной позиции(0,2,4 и тд),то курсор помещается в позицию (x,y\_bites) и выводится на экран соответствующий бит.Потом стирается с помощью функции clear\_screen(int x1, int x2, int y1, int y2)и переменная y\_bites,которая изначально была равна нулю, увеличивается на 1,потом снова стирается и так происходит до тех пор,пока y\_bites не станет равно y-параметру,который принимала функция dynamic\_output(..) при вызове. Вывод четного бита происходит сверху вниз.Если же бит стоит на нечетной позиции(1,3,5 и тд),то происходит тоже самое,только y\_bites=30 и в процессе выполнения программы уменьшается до y.То есть вывод как бы происходит снизу-вверх.

template<class T>

void dynamic\_output(T number, int x = 30, int y = 10)

{

bool bit;

int k = sizeof(T) - 1, i = 0, count = 0, y\_bites = 0;

types<T> u;

u.d = number;

bool ch[8 \* sizeof(T)];

while (k >= 0)

{

i = 7;

while (i >= 0)

{

bit = u.c[k] & (1U << i);

ch[count] = bit;

i--;

count++;

}

k--;

}

count = 8 \* sizeof(T);

i = 0;

while (i < count)

{

bit = ch[i];

if (i % 2 == 0)

{

y\_bites = 0;

while (y\_bites != y)

{

gotoxy(x, y\_bites);

color(0, 11);

cout << bit;

Sleep(3);

gotoxy(x, y\_bites);

cout << ' ';

y\_bites++;

}

gotoxy(x, y\_bites);

color(0, 15);

cout << bit;

}

else

{

y\_bites = 30;

while (y\_bites != y)

{

gotoxy(x, y\_bites);

color(0, 10);

cout << bit;

Sleep(3);

gotoxy(x, y\_bites);

cout << ' ';

y\_bites--;

}

gotoxy(x, y\_bites);

color(0, 15);

cout << bit;

}

i++;

x++;

}}

**Задание №4**

**Текст программы и краткое описание:**

В функцию chart(T number) подается значение переменной number.Далее в переменную типа string s записывается двоичное представление числа number.

Затем идет цикл,который нужен для изменения масштаба графика.В переменную input записываются код клавиши, введенной пользователем.Если это стрелки вправо/влево,то происходит увеличение/уменьшение переменной coefficient,которая задает масштаб, и затем вызывается функция phase\_modulation(s, coefficient);

При нажатии клавиши tab вызывается функция изменения цвета. Там можно выбирать цвет выводимого графика на экран.

template<class T>

void chart(T number)

{

string s;

bool bit;

int k = sizeof(T) - 1, i = 0, count = 0,input=0coefficient=0;

types<T> u;

u.d = number;

while (k >= 0)

{

i = 7;

while (i >= 0)

{

bit = u.c[k] & (1U << i);

if (!bit)

s = s + '0';

else

s = s + '1';

i--;

count++;

}

k--;

}

while (input != \_esc) {

gotoxy(5, 25);

if (input == \_right && coefficient <3) {

clear\_screen(41, 80, 8, 9);

coefficient++;

phase\_modulation(s, coefficient);

}

if (input == \_left && coefficient > 0) {

clear\_screen(41, 80, 8, 9);

coefficient--;

phase\_modulation(s, coefficient);

}

if (input == \_tab)

{

show\_menu\_for\_color();

system("cls");

}

if (input == \_esc)

break;

input = \_getch();

}

system("pause");

}

Далее функция phase\_modulation(s, k) принимает переменную типа string,в которой записано двоичное представление числа number и переменную целого типа k. В зависимости от значения переменной k, присваивается определенное значение переменной count,которая отвечает за масштаб графика.

if (k == 1)

{

count = 0;

}

else if (k == 2) {

count = 3;

}

else if (k == 3) {

count = 7;

}

В цикле пока не будут проверены все биты, выводится график в соответствии масштабу,который выбирает пользователь.

MoveToEx(hDC, x, y, 0)- обновляет текущую позицию указанной точки

LineTo(hDC, x, y )- чертит линию от текущей позиции до, но не включая в нее, указанной точки.

while (i < size) {

MoveToEx(hDC, x, y, 0);

if (ch[i]=='1' && ch[i+1] == '0') {

y += 20;

x += count;

LineTo(hDC, x, y );

}

if (ch[i] == '0' && ch[i + 1] == '1') {

y -= 20;

x += count;

LineTo(hDC, x, y);

}

if(count==0)

{

x += 20 ;

LineTo(hDC, x, y);

}

if (count != 0 && ch[i] == ch[i + 1]) {

//x += 20;

\_asm

{

mov eax,x

mov ebx,20

add eax,ebx

mov x,eax

}

LineTo(hDC, x, y);

}

if(count !=0 && ch[i]!=ch[i+1]){

x += 20 - 2 \* count;

LineTo(hDC, x, y);

}

i++;

}

}

**Описание оставшихся функций:**

int main()-главная функция.

Здесь реализуется выбор типа данных и выбор цвета битов. С помощью клавиш-стрелок вверх и вниз можно выбирать нужный тип данных. Есть переменная счетчик y,которая при нажатии клавиши-стрелки вверх уменьшается на 1,а при нажатии клавиши-стрелки вниз увеличивается на 1. При нажатии enter и вызывается функция change\_type\_data, в которую посылается переменная y.

Если пользователь нажимает клавишу-стрелку вправо, то курсор перескакивает в правый верхний угол экрана и при нажатии enter вызывается функция show\_menu\_for\_color(),позволяющая выбрать цветы битов задолго до выполнения перестановки битов. Затем при нажатии клавиши-стрелки влево курсор перескакивает обратно в левую часть экрана, где можно продолжить выбирать тип данных.

void change\_type\_data(int count)-функция, у которой целочисленный параметр count принимает цифру от 0 до 7.Это цифра обозначает какой тип данных выбрал пользователь. Затем в целочисленную переменную number\_system функция entering\_number\_system() возвращает число от 1 до 128.Это номер системы счисления. Далее в операторе switch(count) вызывается функция, соответствующая определенному значению count. Например, если бы значение count было бы равно 4,то вызвалась бы следующая функция:

\_int(int number\_system,int count)

В эту функцию посылаются значения переменных number\_system и count.

int entering\_number\_system()-функция ввода системы счисления.

Внутри тела бесконечного цикла в переменную целого типа ch записывается код символа с помощью функции getch(),которая возвращает очередной символ, считанный с консоли. Если этот символ является цифрой,то в переменную целого типа x записывается этот символ. Если пользователь нажимает клавишу enter,то функция завершает работу и возвращает значение переменной x.Также с помошью клавишы backspace можно удалять цифры в числе.

template<class T>

T entering\_number(T number,int number\_system,int count)-функция ввода числа.В этой функции в цикле организован посимвольный ввод числа. В переменную string s записывается очередной введенный символ. В зависимости от того, какой тип данных выбрал пользователь можно ввести тот или иной символ. Например:

Если пользователь выбрал тип int , то можно ввести с клавиатуры только символы данной СС. Если же тип double,то помимо символов данной системы счисления можно ввести символ “.”.

При вводе числа на экран одновременно выводятся представление этого числа в данной СС, десятичной СС и двоичной СС. Пользователь может редактировать свой ввод: нажимая клавишу backspace удаляется последняя введенная цифра.

double transfer\_number(string s,int number\_system)-функция, которая переводит введенную строку в число типа double.Параметры этой функции принимают соответственно значение данной строки и значение системы счисления. Внутри цикла есть две ключевые строки, отвечающие за перевод числа:

number\_ten = check\_number(s[i], number\_system);

number = number + number\_ten \* pow(number\_system, k);

template<class T>

T check\_number(T ch,int number\_system)-функция, отвечающая за проверку символов числа в данной СС. Функция принимает переменную типа T и основание системы счисления. Далее идет проверка этого символа на принадлежность к этой СС. Например:

adress = int(ch);

if ((ch >= '0') && (ch <= '9') &&(adress-48)<number\_system)

{

return (adress - 48);

}

Если этот символ цифра и эта цифра меньше, чем основание СС(Например: 7<9),то функция возвращает число целого типа. Если ни одно из условий не выполнилось, функция возвращает -1.

template<class T>

T check\_range(double ch,T number,int number\_system,int count)-проверка диапазона введенного числа. Функция принимает значение числа типа double,число number==0 типа T,основание СС.А в переменную count записывается цифра от 0 до 7,где потом с помощью switch(count)-case определяется какого типа пользователь ввел число. Например тип char может хранить число в диапазоне от 0 до 255.

case 0:

if ((ch <= (255) && ch >= 0))

{

number = static\_cast<T>(ch);

}

else

number = false;

break;

Данный код работает в том случае, если пользователь выбрал работу с типом char.Если введенное число принадлежит заданному промежутку, то из типа double с помощью static\_cast<T>(ch) число переводится в тип char без каких либо потерь точности.

void \_bool(int number\_system,int count)

void \_char(int number\_system,int count)

void short\_int(int number\_system,int count)

void unsigned\_short\_int(int number\_system,int count)

void \_int(int number\_system,int count)

void unsigned\_int(int number\_system,int count)

void \_float(int number\_system,int count)

void \_double(int number\_system,int count)-функции, с помощью которых происходит ввод числа, динамический вывод, перестановка битов и построение графика соответствующего типа данных

template<class T>

T input\_group\_bit(T number, int number\_system)-функция, позволяющая пользователю ввести 2 группы бит для того чтобы впоследствии их переставить.

Внутри цикла в переменную типа int записывается код очередной нажатой клавиши.

ch = \_getch();

Далее есть ряд условий, и в зависимости от того какая была нажата клавиша выполняются определенные действия. Например, пользователь с помощью клавиш-стрелок вправо/влево передвигает курсор в числе.

Этот фрагмент кода выполнится, если пользователь нажмет клавишу-стрелку вправо. Здесь i-последний символ числа,x-текущий.

if (ch == \_right && x < i)

{

x++;

gotoxy(x + 30, 10);

}

Как только он захотел выбрать нужный бит, он должен нажать space.Можно выбирать только подряд идущие биты. Если же пользователь передумал и захотел удалить выбранный бит, то он должен нажать клавишу backspace.Для того, чтобы завершить ввод одной группы бит, нужно нажать enter.

Внутри этой функции вызывается функция swap\_bites, рассмотренная раннее, и возвращается уже измененное число.

number = swap\_bites(number, s1, s2, number\_system);

return number;

void show\_menu\_for\_color()-функция, позволяющая выбрать цвет для группы битов, во время их перестановки.

В теле бесконечного цикла в переменную типа int ch записывается код клавиши введенной пользователем. Если была нажата клавиша enter,то далее вызываются функции getColor(),описанные ранее, которые возвращают текущий цвет рисования битов. Эти значения записываются в переменные целого типа color\_bit\_one и color\_bit\_two.С помощью клавиши esc можно заранее завершить выбор цвета.

{

if (ch == \_enter)

{

system("cls");

gotoxy(0, 0);

color(0, 10);

cout << "цвет 1 бита";

color(0, 15);

color\_bit\_one = getColor();

clear\_screen(30, 33, 3, 14);

gotoxy(0, 0);

color(0, 5);

cout << "цвет 2 бита";

color(0, 15);

color\_bit\_two = getColor();

return;

}

if (ch == \_esc)

{

system("cls");

return;

}

ch = \_getch();

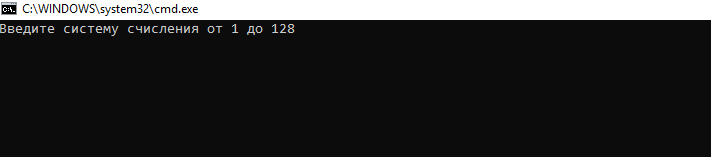
}

**ПРИМЕР ЗАПУСКА ПРОГРАММЫ:**

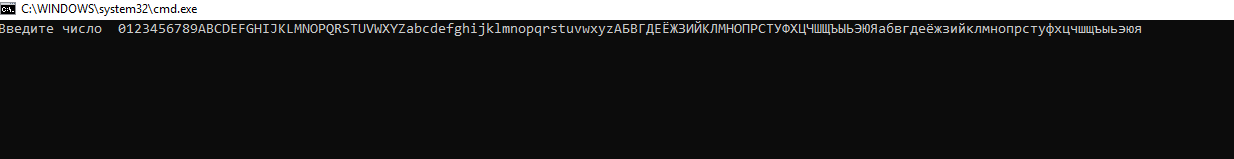
1. При запуске программы открывается меню. С помощью клавиш вверх и вниз можно двигать стрелку, а нажимая enter выбрать соответствующий тип данных.

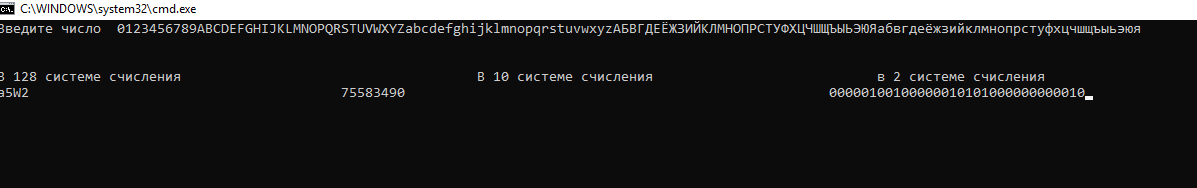


2.Можно выбрать систему счисления от 1 до 128



3. Программа просит ввести число. Во время ввода пользователь может видеть допустимые символы для ввода.

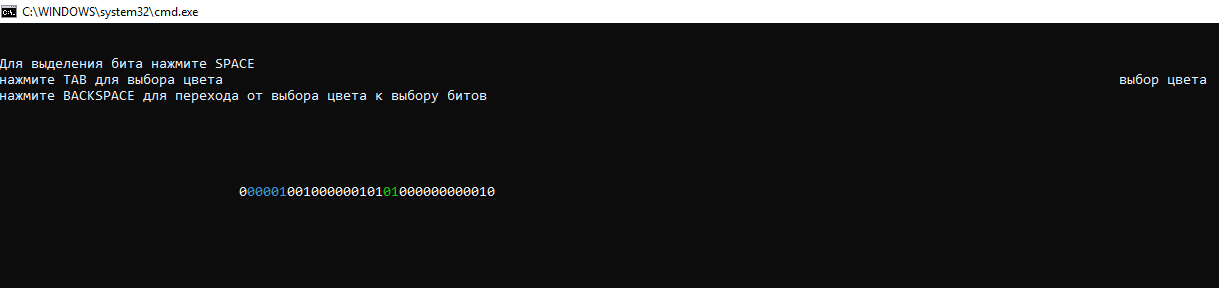


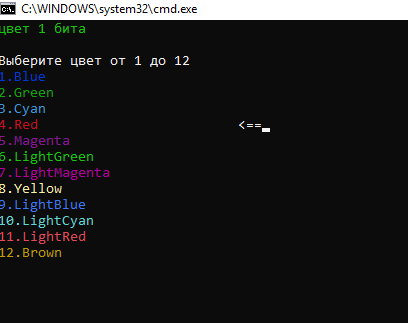
Во время ввода происходит одновременный перевод числа в 10 и 2 СС.

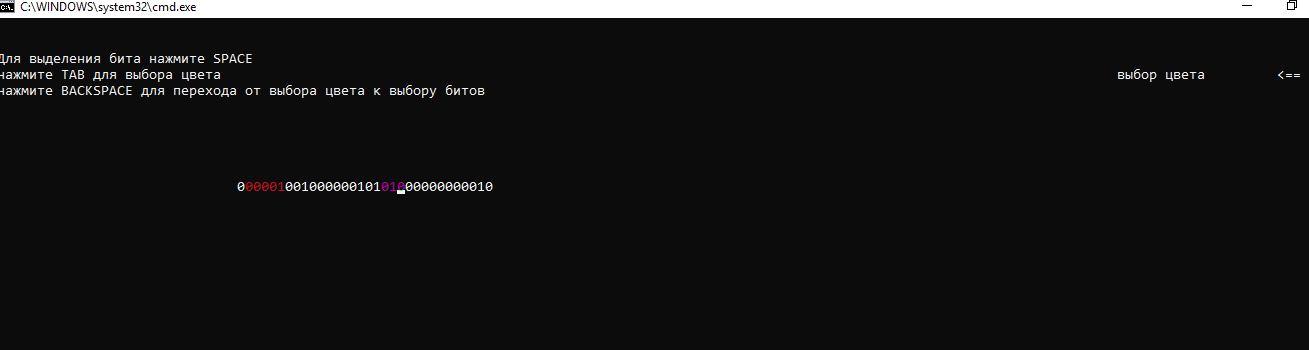
4.После ввода числа программа выдает его двоичное представление. И запрашивает пользователя выбрать группы бит для перестановки. Группы бит

выбираются с помощью клавиш: стрелка влево/вправо, space, backspace, enter.

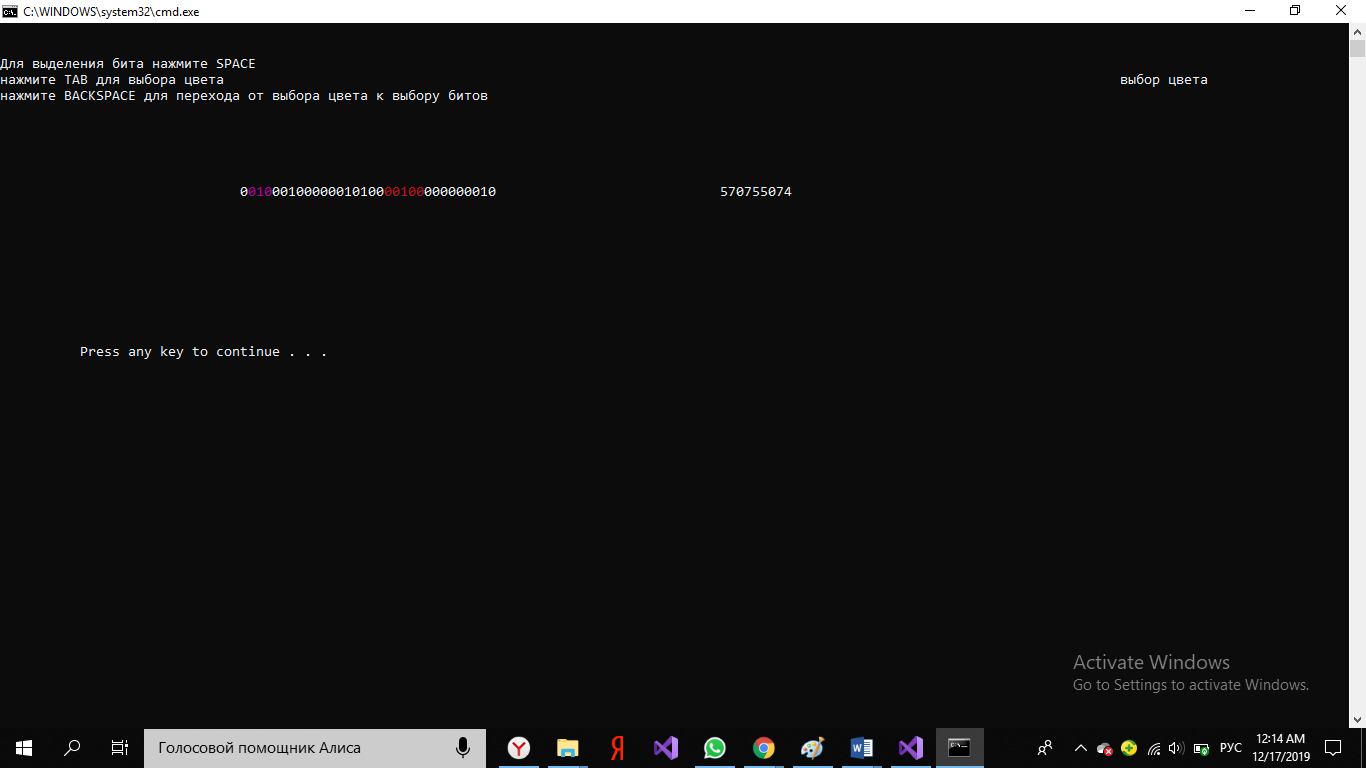
Стрелки используются для перемещения курсора; пробел позволяет выбрать и перекрасить бит, в котором стоит курсор; backspace отменяет ввод бита, то есть перекрашивает его обратно в белый цвет; enter заканчивает ввод группы бит.



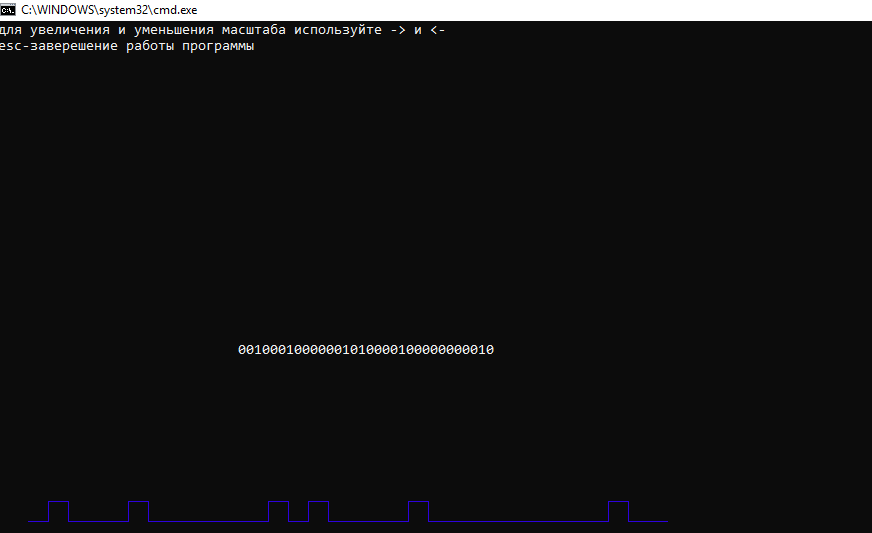
5.Также пользователь может выбрать другой цвет группы бит



6. После нажатия клавишы enter биты переставляются и выводится число в 10 СС



7.После нажатия любой клавишы программа выводит число в 2 СС и график фазовой модуляции.



С помощью стрелок вправо/влево можно менять масштаб:



**Вывод:**

Все задания были представлены в одной программе. В ходе работы был приобретен опыт работы с ассемблером. Это подтверждает наличие ассемблерных вставок. Были также изучены аппаратные средства, используемые при выполнении программы. Структурная схема наглядно показывает: что, где и как расположено в памяти компьютера. Весь код был проверен с помощью отладки и не содержит ошибок.