Курс «Основы программирования»

Власенко Олег Федосович
SimbirSoft

Лекция 7

Polyline/Polygon. Типы данных. Вложенные циклы

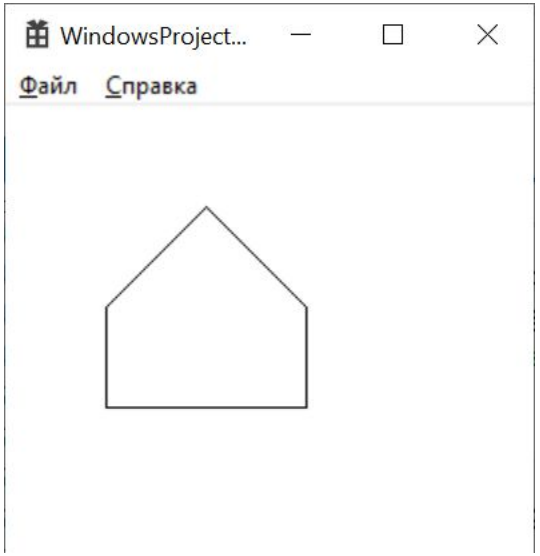
ЛР 12. Использование Polyline и Polygon

ЛР 13. Вложенные циклы

Polygon / Polyline

Polygon

```
POINT pt[5]; // Массив содержит структуры
pt[0].x = 100;
pt[0].y = 50;
pt[1].x = 150;
pt[1].y = 100;
pt[2].x = 150;
pt[2].y = 150;
pt[3].x = 50;
pt[3].y = 150;
pt[4].x = 50;
pt[4].y = 100;
Polygon(hdc, pt, 5);
```

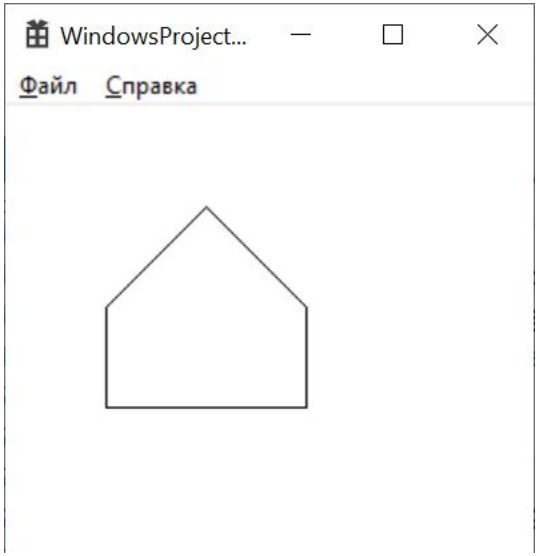


Что такое POINT?

```
// windef.h?
typedef struct tagPOINT
{
    LONG  x;
    LONG  y;
} POINT;
```

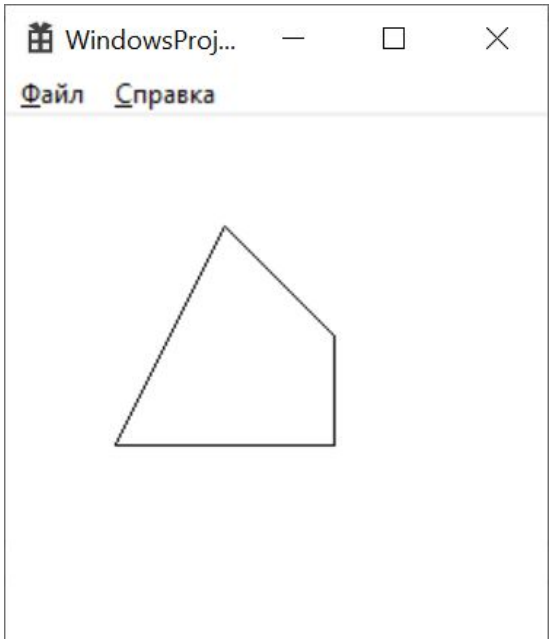
Polygon

```
POINT pt[5]; // Массив содержит структуры
pt[0].x = 100;
pt[0].y = 50;
pt[1].x = 150;
pt[1].y = 100;
pt[2].x = 150;
pt[2].y = 150;
pt[3].x = 50;
pt[3].y = 150;
pt[4].x = 50;
pt[4].y = 100;
Polygon(hdc, pt, 5);
```



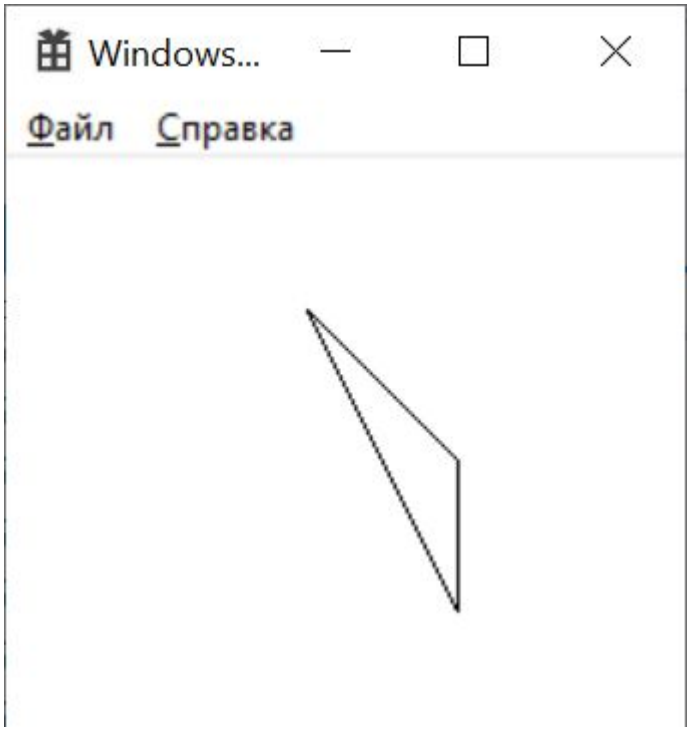
Polygon

```
POINT pt[5];
pt[0].x = 100;
pt[0].y = 50;
pt[1].x = 150;
pt[1].y = 100;
pt[2].x = 150;
pt[2].y = 150;
pt[3].x = 50;
pt[3].y = 150;
pt[4].x = 50;
pt[4].y = 100;
Polygon(hdc, pt, 4);
```



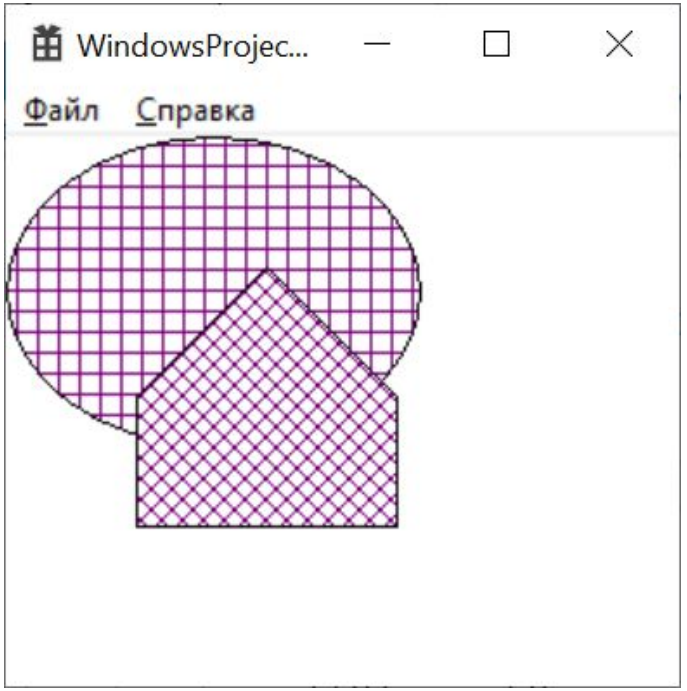
Polygon

```
POINT pt[5];
pt[0].x = 100;
pt[0].y = 50;
pt[1].x = 150;
pt[1].y = 100;
pt[2].x = 150;
pt[2].y = 150;
pt[3].x = 50;
pt[3].y = 150;
pt[4].x = 50;
pt[4].y = 100;
Polygon(hdc, pt, 3);
```



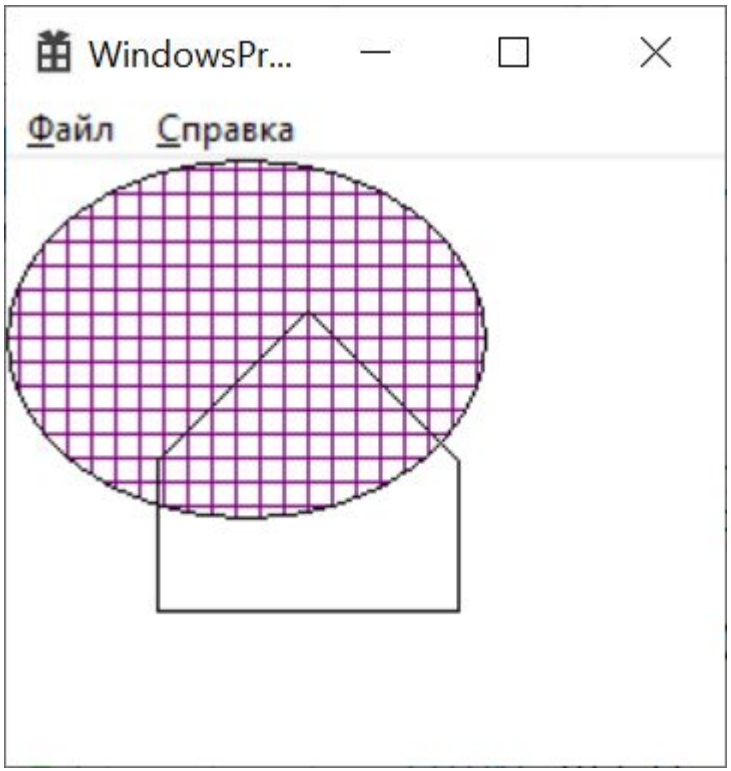
Polygon

```
HBRUSH hBrush = CreateHatchBrush(HS_CROSS, RGB(128, 0, 128));
SelectObject(hdc, hBrush);
Ellipse(hdc, 0, 0 , 160, 120);
HBRUSH hBrush2 = CreateHatchBrush(HS_DIAGCROSS, RGB(128, 0, 128));
SelectObject(hdc, hBrush2);
POINT pt[5];
pt[0].x = 100;
pt[0].y = 50;
pt[1].x = 150;
pt[1].y = 100;
pt[2].x = 150;
pt[2].y = 150;
pt[3].x = 50;
pt[3].y = 150;
pt[4].x = 50;
pt[4].y = 100;
Polygon(hdc, pt, 5);
```



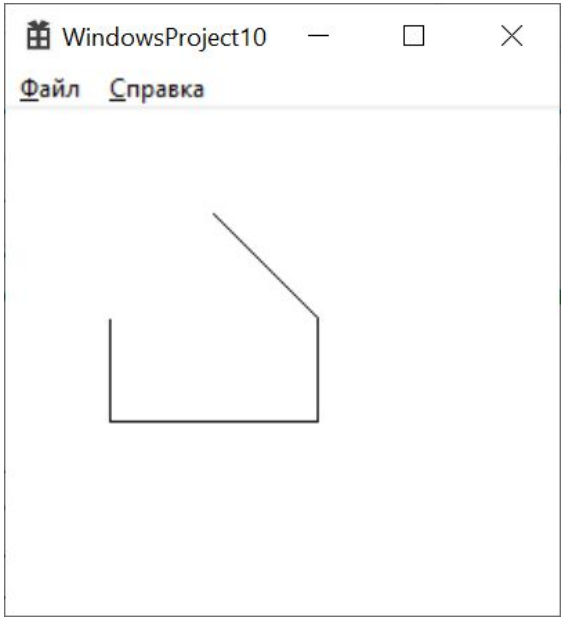
Polygon – прозрачная кисть

```
HBRUSH hBrush = CreateHatchBrush(HS_CROSS, RGB(128, 0, 128));
SelectObject(hdc, hBrush);
Ellipse(hdc, 0, 0 , 160, 120);
SelectObject(hdc, GetStockObject(NULL_BRUSH));
POINT pt[5];
pt[0].x = 100;
pt[0].y = 50;
pt[1].x = 150;
pt[1].y = 100;
pt[2].x = 150;
pt[2].y = 150;
pt[3].x = 50;
pt[3].y = 150;
pt[4].x = 50;
pt[4].y = 100;
Polygon(hdc, pt, 5);
```



Polyline

```
POINT pt[5];
pt[0].x = 100;
pt[0].y = 50;
pt[1].x = 150;
pt[1].y = 100;
pt[2].x = 150;
pt[2].y = 150;
pt[3].x = 50;
pt[3].y = 150;
pt[4].x = 50;
pt[4].y = 100;
Polyline(hdc, pt, 5);
```

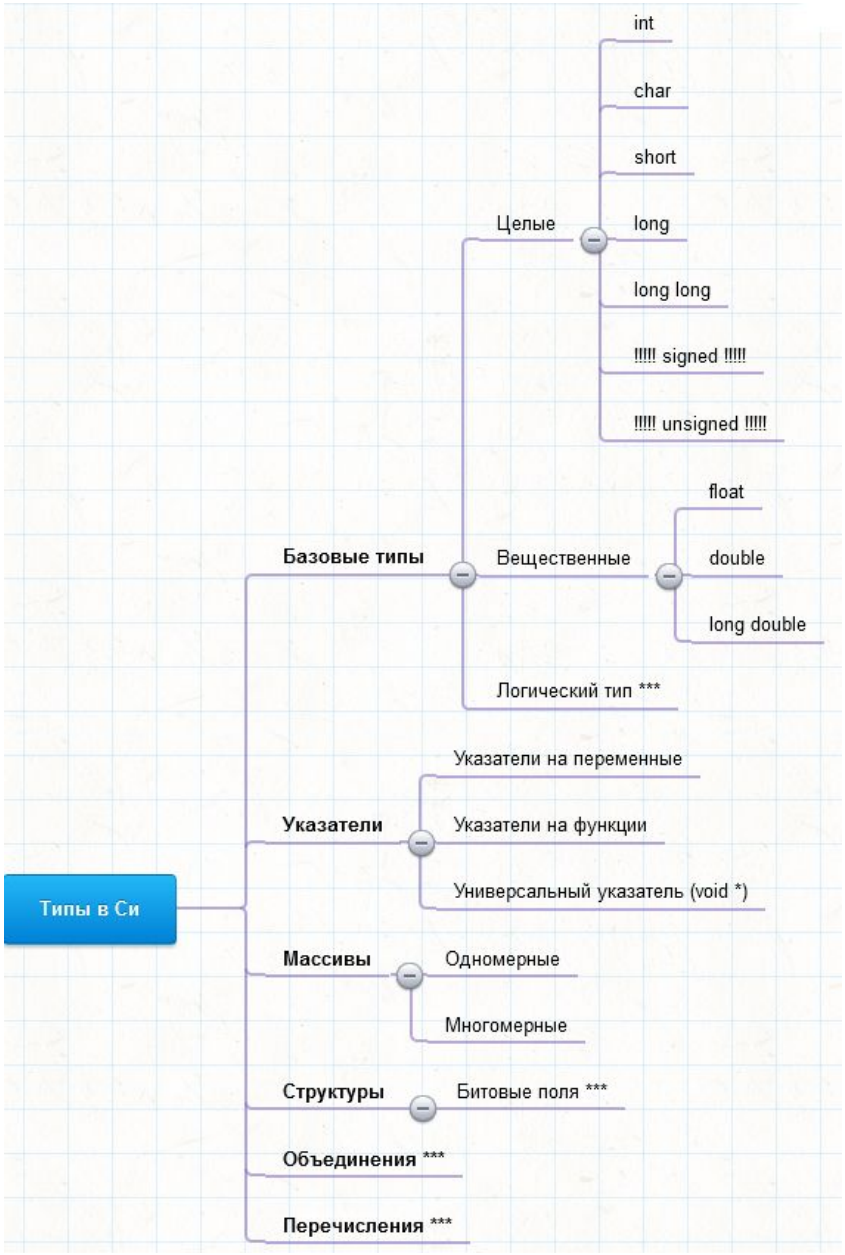


Источники информации

- http://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm - **Рисование геометрических фигур**

Типы данных в Си

Какие типы есть в Си?



Где прочитать про типы данных в Си?

С.Ю. Курсков Введение в язык Си

Типы данных - https://dfe.petrsu.ru/koi/posob/c/c.htm#g1.2

Указатели и операции с ними - https://dfe.petrsu.ru/koi/posob/c/c.htm#g2.3

Массивы - https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.1

Структуры - https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.3

Битовые поля - https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.5

Объединение (union) - https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.6

Перечислимый тип данных - https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.7

Указатели на функции - https://dfe.petrsu.ru/koi/posob/c/c.htm#g4.3

***Система типов Си -*
https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2_%D0%A1%D0%B8

*****Б. Керниган, Д. Ритчи Язык программирования Си*

Типы и размеры данных - http://givi.olnd.ru/kr2/02.html#c0202

Программное обеспечение

https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5

Структура компьютера

https://www.zaurtl.ru/UkVT/UKVT3.html

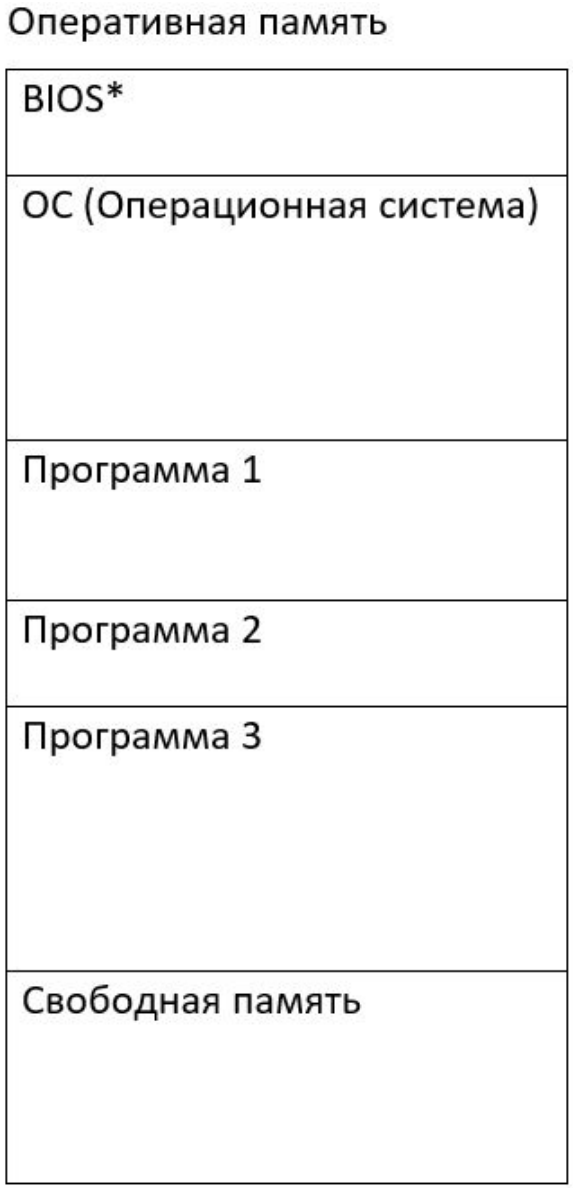
Оперативная память

https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C

Операти́вная па́мять (англ. *Random Access Memory, RAM* — память с произвольным доступом) — в большинстве случаев энергозависимая часть системы компьютерной памяти, в которой во время работы компьютера хранится выполняемый машинный код (программы), а также входные, выходные и промежуточные данные, обрабатываемые процессором.

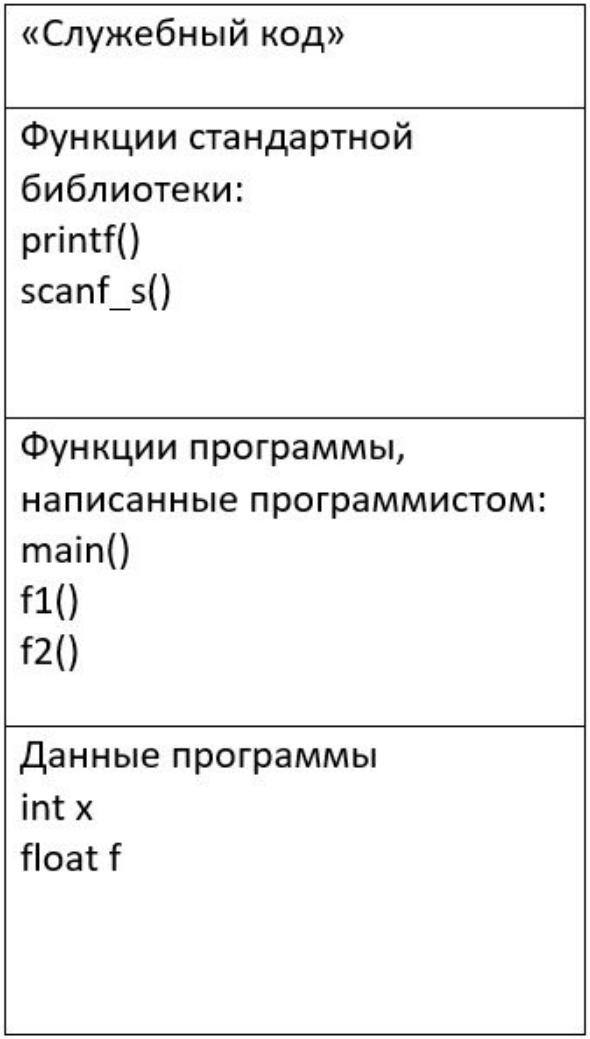
Модули ОЗУ для ПК

Оперативная память во время работы компьютера



Оперативная память доступная программе

Си программа в оперативной памяти



Размещение переменных в оперативной памяти

адрес	переменная
0	a
1	
2	
3	
4	b
5	
6	c
7	
8	d1
9	d2
10	d3
11	d4
12	e
13	
14	
15	
...	
12923456	
12923457	

```
long a;
short b;
unsigned short c;
char d1, d2, d3, d4;
float e;
```

Шестнадцатиричная система счисления (16СС)

10 СС	16 СС	16 СС в Си
0	0	0x0
1	1	0x1
2	2	0x2
3	3	0x3
4	4	0x4
5	5	0x5
6	6	0x6
7	7	0x7
8	8	0x8
9	9	0x9
10	A	0xA
11	B	0xB
12	C	0xC
13	D	0xD
14	E	0xE
15	F	0xF

Размещение переменных в оперативной памяти

адрес	переменная
00000000	a
00000001	
00000002	
00000003	
00000004	b
00000005	
00000006	c
00000007	
00000008	d1
00000009	d2
0000000A	d3
0000000B	d4
0000000C	e
0000000D	
0000000E	
0000000F	
...	
00C53240	
00C53241	
...	
FFFFFFFE	
FFFFFFFF	

```
long a;
short b;
unsigned short c;
char d1, d2, d3, d4;
float e;
```

* ОБЫЧНО адреса в памяти отображаются в 16СС

Целые типы в Си

Тип	*sizeof*	*Количество бит*	*MIN*	*MAX*
char	1	8	-128 ***	+127 ***
short	2	16	−32768	+32767
long	4	32	−2 147 483 648	+2 147 483 647
long long	8	64	−9 223 372 036 854 775 808	+9 223 372 036 854 775 807
int	???	???	???	???

“**int**” – это либо “**short**”, либо “**long**”
в MS VS **int** = **long**

sizeof (тип) или sizeof (переменная) – сколько памяти в байтах занимает переменная

“signed” VS “unsigned”

Тип	*sizeof*	*Количество бит*	*MIN*	*MAX*
signed char	1	8	-128	+127
char	1	8	?????	?????
unsigned char	1	8	0	+255
signed short	2	16	−32768	+32767
unsigned short	2	16	0	+65535
signed long	4	32	−2 147 483 648	+2 147 483 647
unsigned long	4	32	0	+4 294 967 295
signed long long	8	64	−9 223 372 036 854 775 808	+9 223 372 036 854 775 807
unsigned long long	8	64	0	+18 446 744 073 709 551 615

“char” не определен явно ни как “signed” ни как “unsigned”.
Знаковый он или беззнаковый зависит от компилятора.

Все целые типы Си (все синонимы)

Тип	*Синонимы*
signed char	signed char
char	char
unsigned char	unsigned char
signed short	short | short int | signed short | signed short int
unsigned short	unsigned short | unsigned short int
signed long	long | long int | signed long | signed long int
unsigned long	unsigned long | unsigned long int
signed long long	long long | long long int | signed long long | signed long long int
unsigned long long	unsigned long long | unsigned long long int
int	int | signed | signed int
unsigned int	unsigned | unsigned int

“char” не определен явно ни как “signed” ни как “unsigned”.
Знаковый он или беззнаковый зависит от компилятора.

“int” это либо “signed short” либо “signed long”
“unsigned int” это либо “unsigned short” либо “unsigned long”

Использование целых типов

```
void main() {
    printf("main() start!\n");

    unsigned long long  l = 1;
    int i = 0;
    while (l > 0) {
        printf("<%llu (%d)>\n", l, i);
        l *= 2;
        i += 1;
    }
    l--;
    printf("<!!!!%llu (%d)!!!!>\n", l, i);

    printf("main() finish!\n");
}
```

Использование целых типов

```
void main() {
    printf("main() start!\n");

    unsigned long long  l = 1;
    int i = 0;
    while (l > 0) {
        printf("<%llu (%d)>\n", l, i);
        l *= 2;
        i += 1;
    }
    l--;
    printf("<!!!!%llu (%d)!!!!>\n", l,

    printf("main() finish!\n");
}
```

```
main() start!
<1 (0)>
<2 (1)>
<4 (2)>
<8 (3)>
<16 (4)>
<32 (5)>
<64 (6)>
<128 (7)>
<256 (8)>
<512 (9)>
<1024 (10)>
<2048 (11)>
<4096 (12)>
<8192 (13)>
<16384 (14)>
<32768 (15)>
<65536 (16)>
<131072 (17)>
<262144 (18)>
<524288 (19)>
<1048576 (20)>
<2097152 (21)>
<4194304 (22)>
<8388608 (23)>
<16777216 (24)>
<33554432 (25)>
<67108864 (26)>
<134217728 (27)>
<268435456 (28)>
<536870912 (29)>
<1073741824 (30)>
<2147483648 (31)>
<4294967296 (32)>
<8589934592 (33)>
<17179869184 (34)>
<34359738368 (35)>
<68719476736 (36)>
<137438953472 (37)>
<274877906944 (38)>
<549755813888 (39)>
<1099511627776 (40)>
<2199023255552 (41)>
<4398046511104 (42)>
<8796093022208 (43)>
<17592186044416 (44)>
<35184372088832 (45)>
<70368744177664 (46)>
<140737488355328 (47)>
<281474976710656 (48)>
<562949953421312 (49)>
<1125899906842624 (50)>
<2251799813685248 (51)>
<4503599627370496 (52)>
<9007199254740992 (53)>
<18014398509481984 (54)>
<36028797018963968 (55)>
<72057594037927936 (56)>
<144115188075855872 (57)>
<288230376151711744 (58)>
<576460752303423488 (59)>
<1152921504606846976 (60)>
<2305843009213693952 (61)>
<4611686018427387904 (62)>
<9223372036854775808 (63)>
<!!!!18446744073709551615 (64)!!!!>
main() finish!
```

Использование целых типов

```
void main() {
    printf("main() start!\n");

    short  s = 1;
    int i = 0;
    while (s > 0) {
        printf("<%d (%d)>\n", s, i);
        s *= 2;
        i += 1;
    }
    s--;
    printf("<!!!!%d (%d)!!!!>\n", s, i);

    printf("main() finish!\n");
}
```

Использование целых типов

```
void main() {
    printf("main() start!\n");

    short  s = 1;
    int i = 0;
    while (s > 0) {
        printf("<%d (%d)>\n", s, i);
        s *= 2;
        i += 1;
    }
    s--;
    printf("<!!!!%d (%d)!!!!>\n", s, i);

    printf("main() finish!\n");
}
```

```
main() start!
<1 (0)>
<2 (1)>
<4 (2)>
<8 (3)>
<16 (4)>
<32 (5)>
<64 (6)>
<128 (7)>
<256 (8)>
<512 (9)>
<1024 (10)>
<2048 (11)>
<4096 (12)>
<8192 (13)>
<16384 (14)>
<!!!!32767 (15)!!!!>
main() finish!
```

Вещественные типы в Си

Тип	*sizeof*	*Количество бит*	*MIN*	*MAX*
float	4	32	3.4E-38	3.4E+38
double	8	64	1.7E-308	1.7E+308
long double	8 | 10	64 | 80	???	???

“long double” – либо 8 байтовое число, совпадающее с double, либо более точный 10 байтовый формат – если он реализован в системе.

в MS VS “long double” = “double”

Использование вещественных типов

```
void main() {
    printf("main() start!\n");

    float f = 1.23456789012345678901234567890;
    int i = 0;
    while (i < 1000) {

        printf("<%60.30f (%d)>\n", f, i);
        f *= 10;
        i += 1;
    }

    printf("sizeof: f = %d,   i = %d", sizeof(f),
sizeof(i));

    printf("main() finish!\n");
}
```

Использование вещественных типов

```
void main() {
    printf("main() start!\r

    float f = 1.23456789012
    int i = 0;
    while (i < 1000) {

        printf("<%60.30f (%
        f *= 10;
        i += 1;
    }

    printf("sizeof: f = %d,
sizeof(i));

    printf("main() finish!\
}
```

Использование вещественных типов

```
void main() {
    printf("main() start!\n

    float f = 1.234567890123
    int i = 0;
    while (i < 1000) {

        printf("<%60.30f (%
        f *= 10;
    }

    printf("main() finish!\n
}
```

Консоль отладки Microsoft Visual Studio

```
main() start!
                                                1.234567880630493164062500000000 (0)>
                                               12.345678329467773437500000000000 (1)>
                                              123.456787109375000000000000000000 (2)>
                                             1234.567871093750000000000000000000 (3)>
                                            12345.678710937500000000000000000000 (4)>
                                           123456.789062500000000000000000000000 (5)>
                                          1234567.875000000000000000000000000000 (6)>
                                         12345679.000000000000000000000000000000 (7)>
                                        123456792.000000000000000000000000000000 (8)>
                                       1234567936.000000000000000000000000000000 (9)>
                                      12345679872.000000000000000000000000000000 (10)>
                                     123456798720.000000000000000000000000000000 (11)>
                                    1234567954432.000000000000000000000000000000 (12)>
                                   12345680068608.000000000000000000000000000000 (13)>
                                  123456804880384.000000000000000000000000000000 (14)>
                                 1234568082358272.000000000000000000000000000000 (15)>
                                12345680555147264.000000000000000000000000000000 (16)>
                               123456807698956288.000000000000000000000000000000 (17)>
                              1234568076989562880.000000000000000000000000000000 (18)>
                                                  000000000000000000000000000000 (19)>
                                       inf (993)> 000000000000000000000000000000 (20)>
                                       inf (994)> 000000000000000000000000000000 (21)>
                                       inf (995)> 000000000000000000000000000000 (22)>
                                       inf (996)> 000000000000000000000000000000 (23)>
                                       inf (997)> 000000000000000000000000000000 (24)>
                                       inf (998)> 000000000000000000000000000000 (25)>
                                       inf (999)> 000000000000000000000000000000 (26)>
                                                  000000000000000000000000000000 (27)>
                                                  000000000000000000000000000000 (28)>
                                                  000000000000000000000000000000 (29)>
                                                  000000000000000000000000000000 (30)>
<1234568224281893112667951228518 4.000000000000000000000000000000 (31)>
<1234568200103376720375367734394 88.000000000000000000000000000000 (32)>
<1234568200103376720375367734394 880.000000000000000000000000000000 (33)>
<1234568138206374756106353989438 6688.000000000000000000000000000000 (34)>
<1234568088688773184691142993473 69984.000000000000000000000000000000 (35)>
<1234568088688773184691142993473 699840.000000000000000000000000000000 (36)>
<1234568088688773184691142993473 6998400.000000000000000000000000000000 (37)>
<1234568063335761180126554963539 63573248.000000000000000000000000000000 (38)>
                                                                      inf (39)>
                                                                      inf (40)>
                                                                      inf (41)>
                                                                      inf (42)>
                                                                      inf (43)>
<
<
<
<
<
<
<
sizeof: f = 4,   i = 4main() finish!
```

Использование вещественных типов

```
void main() {
    printf("main() start!\n");

    float f = 1.23456789012345678901234567890;
    int i = 0;
    while (i < 1000) {
        print
        f *= 
        i += 
    }

    printf("s
sizeof(i));

    printf("m
}
```

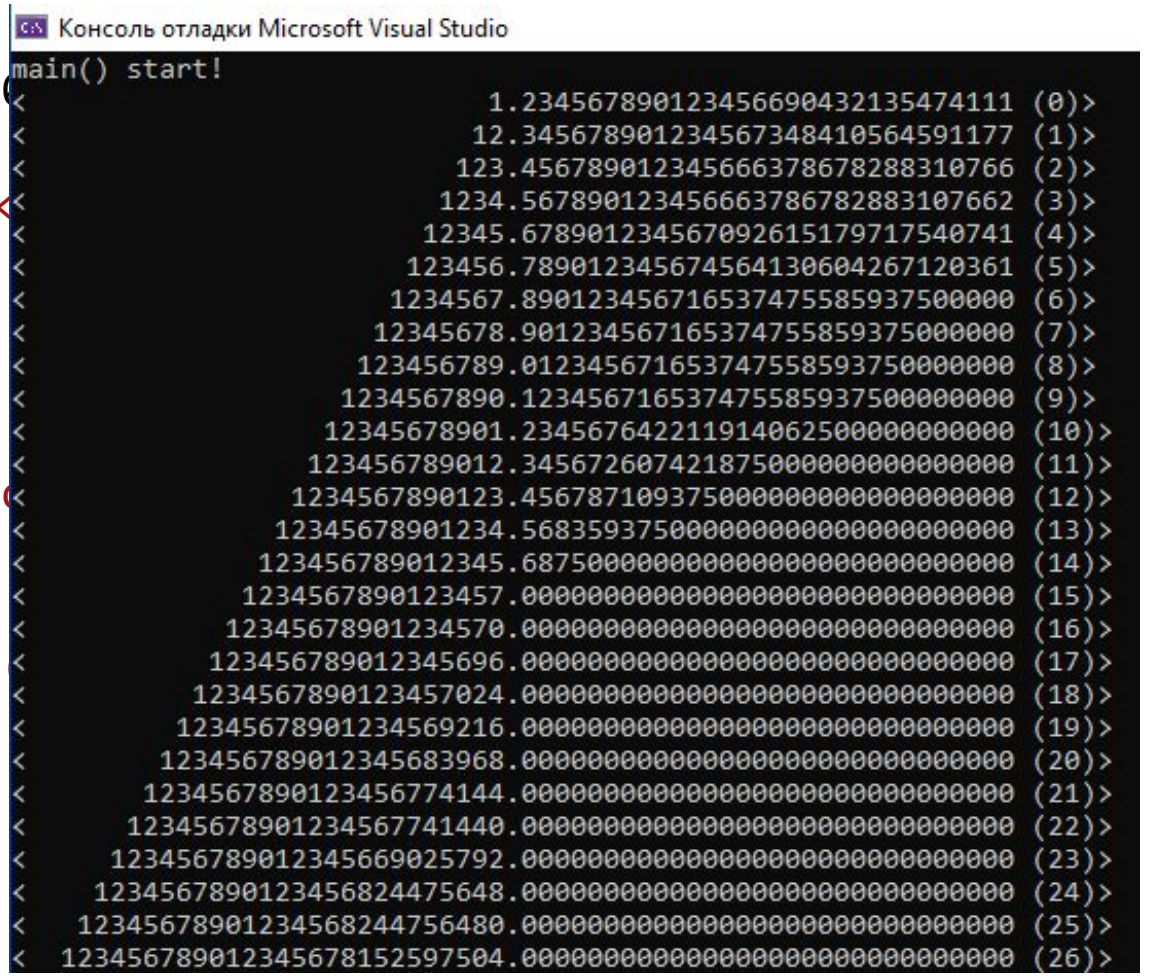
```
main() start!
<                        1.23456788063049316406250000000000 (0)>
<                       12.34567832946777343750000000000000 (1)>
<                      123.45678710937500000000000000000000 (2)>
<                     1234.56787109375000000000000000000000 (3)>
<                    12345.67871093750000000000000000000000 (4)>
<                   123456.78906250000000000000000000000000 (5)>
<                  1234567.87500000000000000000000000000000 (6)>
<                 12345679.00000000000000000000000000000000 (7)>
<                123456792.00000000000000000000000000000000 (8)>
<               1234567936.00000000000000000000000000000000 (9)>
<              12345679872.00000000000000000000000000000000 (10)>
<             123456798720.00000000000000000000000000000000 (11)>
<            1234567954432.00000000000000000000000000000000 (12)>
<           12345680068608.00000000000000000000000000000000 (13)>
<          123456804880384.00000000000000000000000000000000 (14)>
<         1234568082358272.00000000000000000000000000000000 (15)>
<        12345680555147264.00000000000000000000000000000000 (16)>
<       123456807698956288.00000000000000000000000000000000 (17)>
<      1234568076989562880.00000000000000000000000000000000 (18)>
```

Использование вещественных типов

```
void main() {
    printf("main() start!\n");

    double f = 1.23456789012345678901234567890;
    int i = 0;
    while (i < 1000) {

        printf("<%60.30f (%d)>\n", f, i);
        f *= 10;
        i += 1;
    }

    printf("sizeof: f = %d,   i = %d", sizeof(f),
sizeof(i));

    printf("main() finish!\n");
}
```

Использование вещественных типов

```
void main() {
    printf("main() start!\n");

    double f = 1.23456789012345678901234567890;
    int i = 0;
    while (i < 1

        printf("<
        f *= 10;
        i += 1;
    }

    printf("size
sizeof(i));

    printf("main
}
```



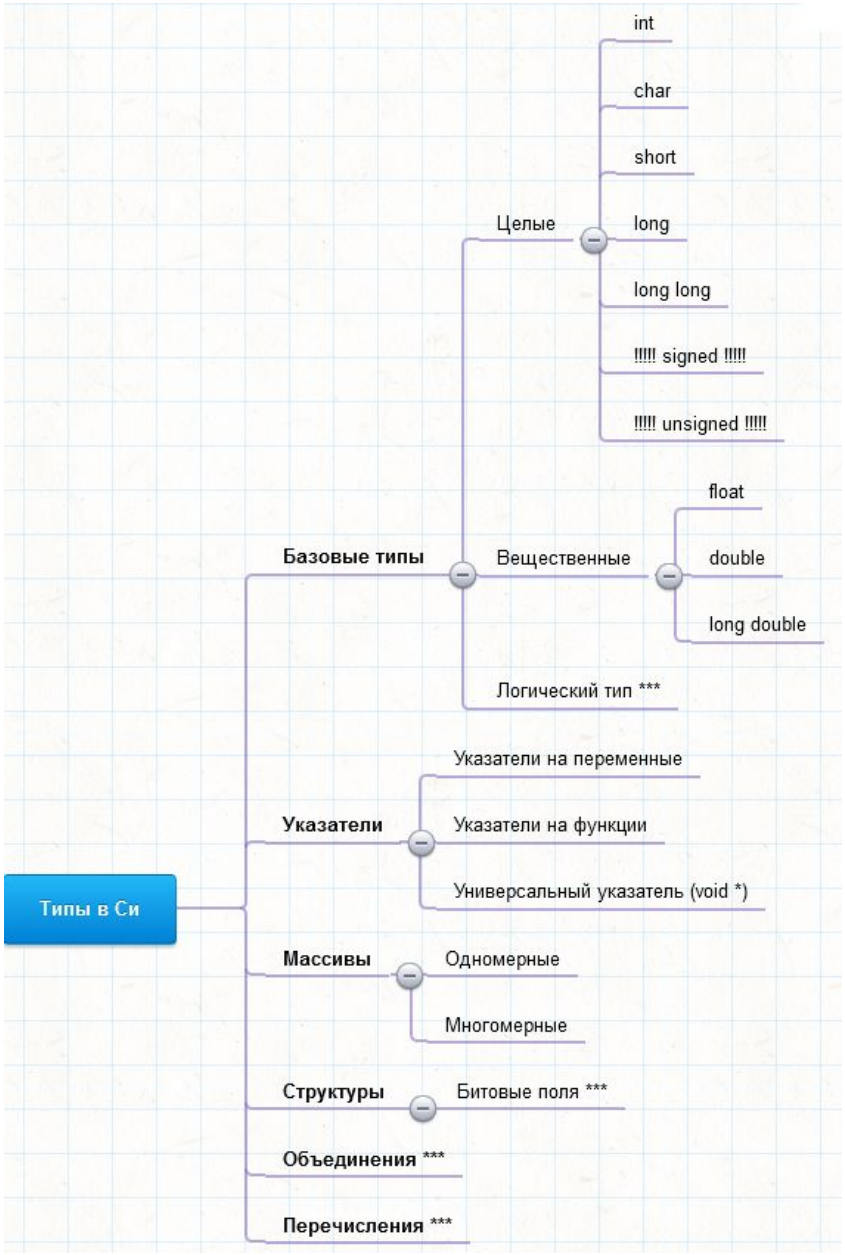
Использование вещественных типов

```
void main() {
    printf("main() start!\n");

    double f = 1.23456789012345678901234567890;
    int i = 0;
    while (i < 1
        printf("<
        f *= 10;
        i += 1;
    }
```

Консоль отладки Microsoft Visual Studio

```
main() start!
<                          1.23456789012345669043213547411 (0)>
<                         12.34567890123456734841056459117 (1)>
<                        123.45678901234566637867828831076 (2)>
<                       1234.56789012345666378678288310766 (3)>
<                      12345.67890123456709261517971754074 (4)>
<                     123456.78901234567456413060426712036 (5)>
<                    1234567.89012345671653747558593750000 (6)>
<                   12345678.90123456716537475585937500000 (7)>
<                  123456789.01234567165374755859375000000 (8)>
<                 1234567890.12345671653747558593750000000 (9)>
<                12345678901.23456764221191406250000000000 (10)>
<               123456789012.34567260742187500000000000000 (11)>
<1234567890123456866867611213947175898109806529786119721396421935041843529606903632285518721018537788144392248029909653009584066179838270890340365865530552805702198549215923971446548580475008465109129516449003594889560850458383908788807927098675944325689239562829706476048609470014565806172822160558807580672.00000000000000000000000000000000 (306)
>
<1234567890123456742127591868027187669586511964983709342180718157825747185697700979959775479208283676584519404740178334925681138417807318142062991590930949009503961156413960201446194021032153682763021926176316895973726053924388659205632273653621545020135785284826946631853552790279813203525416208662306969601792.00000000000000000000000000000000 (307)>
<1234567890123456742127591868027187669586511964983709342180718157825747185697700979959775479208283676584519404740178334925681138417807318142062991590930949009503961156413960201446194021032153682763021926176316895973726053924388659205632273653621545020135785284826946631853552790279813203525416208662306969601792.00000000000000000000000000000000 (308)>
<                                                    inf (309)>
<                                                    inf (310)>
<  12345678901234567815259750 4.00000000000000000000000000000000 (26)>
```

Какие типы есть в Си?



Указатели в Си

Указатели - это переменные, показывающие место или адрес памяти, где расположены другие объекты (переменные, функции и др.).
Так как указатель содержит адрес некоторого объекта, то через него можно обращаться к этому объекту.

Унарная операция & дает адрес объекта, поэтому оператор
y = &x;
присваивает адрес переменной x переменной y.

Унарная операция * воспринимает свой операнд как адрес некоторого объекта и использует этот адрес для выборки содержимого, поэтому оператор
z = *y;
присваивает z значение переменной, записанной по адресу y.

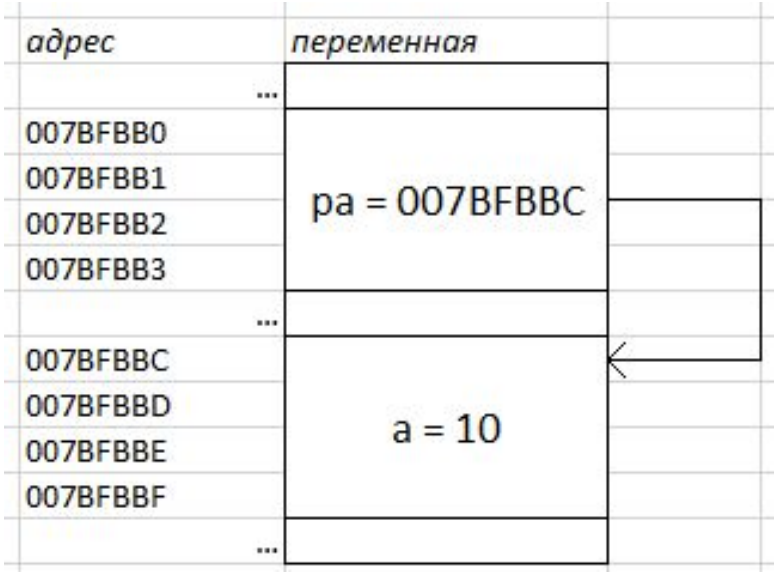
Если
y = &x;
z = *y;
то z = x.
https://dfe.petrsu.ru/koi/posob/c/c.htm#g2.3

Указатели определяются в Си так: *ТИП * имя_указателя;*
Например **int *y; // y – указатель на int**

Пример №1 работы с указателем

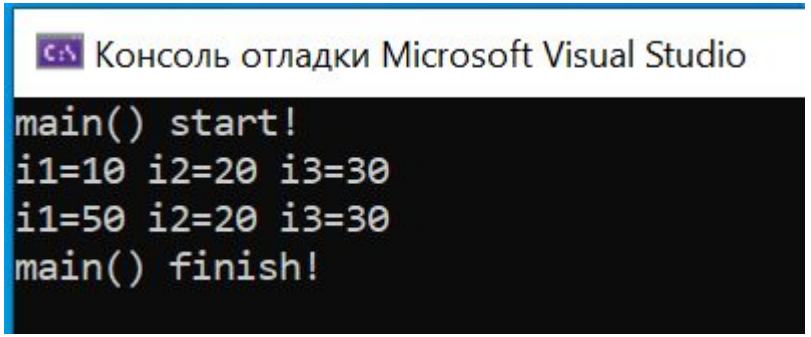
```c
#include <stdio.h>

void main() {
	printf("main() start!\n");

	int a;
	int* pa;

	a = 10;
	pa = &a;

	printf("val: a=%d *pa=%d pa=%p\n", a, *pa, pa);
	printf("ptr: &a=%p &pa=%p\n", &a, &pa);
	printf("sizeof: %d %d\n", sizeof a, sizeof
(pa));

	*pa = 30;

	printf("val: a=%d *pa=%d pa=%p\n", a, *pa, pa);

	printf("main() finish!\n");
}
```

```
Консоль отладки Microsoft Visual Studio
main() start!
val: a=10 *pa=10 pa=007BFBBC
ptr: &a=007BFBBC &pa=007BFBB0
sizeof: 4 4
val: a=30 *pa=30 pa=007BFBBC
main() finish!
```



Пример №2 работы с указателем

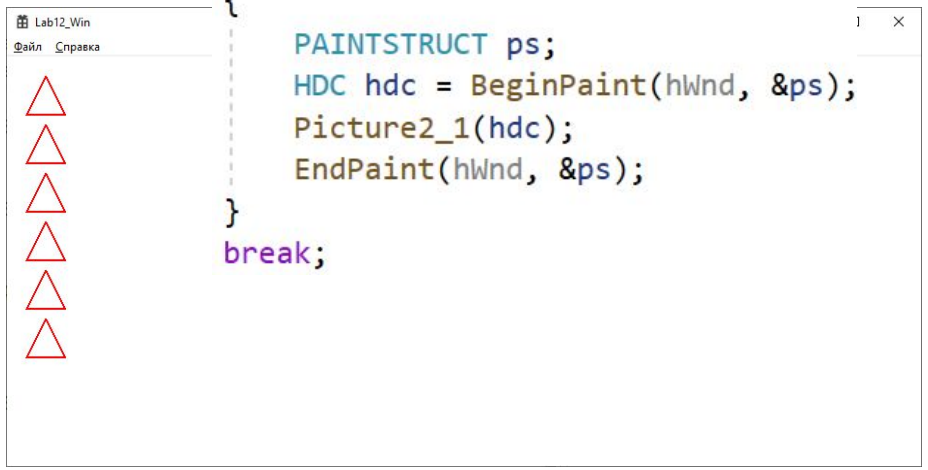
```c
#include <stdio.h>

// В функции f вычисляется сумма a + b
// Результат вычисления помещается по адресу,
// заданному в аргументе ps
void f(int *ps, int a, int b) {
    *ps = a + b;
}

void main() {
    printf("main() start!\n");

    int i1, i2, i3;

    i1 = 10;
    i2 = 20;
    i3 = 30;

    printf("i1=%d i2=%d i3=%d\n", i1, i2, i3);

    f(&i1, i2, i3);

    printf("i1=%d i2=%d i3=%d\n", i1, i2, i3);

    printf("main() finish!\n");
}
```



Пример №3 работы с указателем (1)

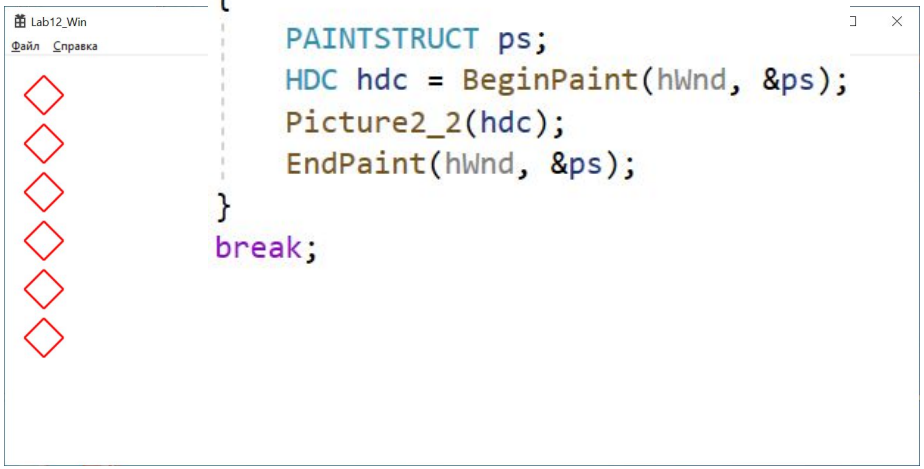
```
void Picture2_1(HDC hdc) {
    int x = 40;
    int y = 40;
    int i = 0;
    while (i < 6) {
        Image1(hdc, x, y);
        y += 50;
        i++;
    }
}
```

```
case WM_PAINT:
{
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hWnd, &ps);
    Picture2_1(hdc);
    EndPaint(hWnd, &ps);
}
break;
```




```
void Picture2_2(HDC hdc) {
    int x = 40;
    int y = 40;
    int i = 0;
    while (i < 6) {
        Image2(hdc, x, y);
        y += 50;
        i++;
    }
}
```

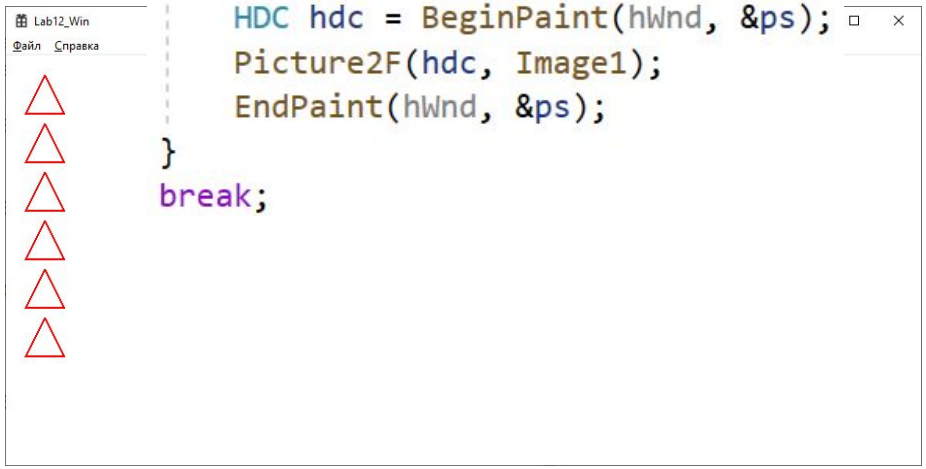
```
case WM_PAINT:
{
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hWnd, &ps);
    Picture2_2(hdc);
    EndPaint(hWnd, &ps);
}
break;
```



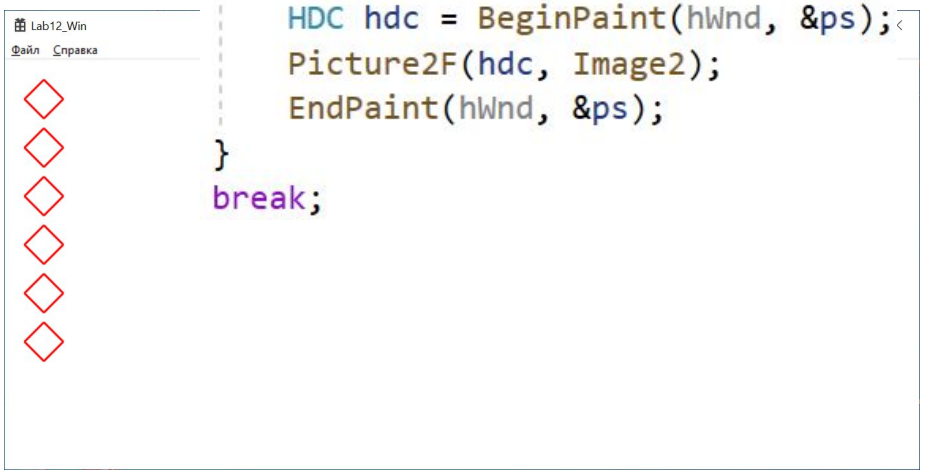
Пример №3 работы с указателем (2)

```
void Picture2F(HDC hdc, void (*pfImage)(HDC hdc, int cx, int cy)) {
    int x = 40;
    int y = 40;
    int i = 0;
    while (i < 6) {
        pfImage(hdc, x, y);
        y += 50;
        i++;
    }
}
```

```
case WM_PAINT:
{
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hWnd, &ps);
    Picture2F(hdc, Image1);
    EndPaint(hWnd, &ps);
}
break;
```




```
case WM_PAINT:
{
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hWnd, &ps);
    Picture2F(hdc, Image2);
    EndPaint(hWnd, &ps);
}
break;
```



Массивы

https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.1

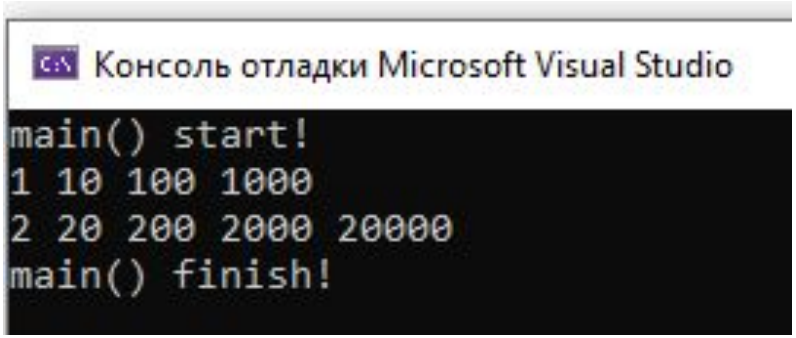
Массив состоит из элементов одного и того же типа.
Ко всему массиву целиком можно обращаться по имени. Кроме того, можно выбирать любой элемент массива. Для этого необходимо задать индекс, который указывает на его относительную позицию. Если массив объявлен, то к любому его элементу можно обратиться следующим образом: указать имя массива и индекс элемента в квадратных скобках.
Массивы определяются так же, как и переменные:

```
int a[100];
char b[20];
float d[50];
```

В первой строке объявлен массив a из 100 элементов целого типа: a[0], a[1], ..., a[99] (индексация всегда начинается с нуля). Во второй строке элементы массива b имеют тип char, а в третьей - float.

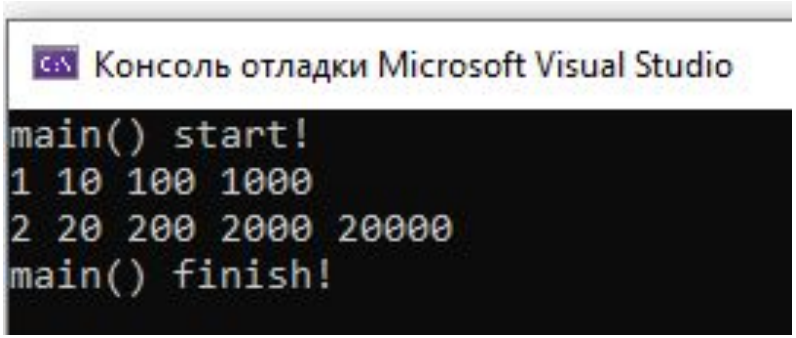
Массивы – пример 1

```c
#include <stdio.h>

void main() {
	printf("main() start!\n");

	int a[4];
	a[0] = 1;
	a[1] = 10;
	a[2] = 100;
	a[3] = 1000;

	printf("%d %d %d %d\n", a[0], a[1], a[2], a[3]);

	int b[5] = {2, 20, 200, 2000, 20000};
	printf("%d %d %d %d %d\n", b[0], b[1], b[2], b[3], b[4]);

	printf("main() finish!\n");
}
```



Массивы – пример 2

```c
#include <stdio.h>
void main() {
	printf("main() start!\n");
	int a[4];
	a[0] = 1;
	a[1] = 10;
	a[2] = 100;
	a[3] = 1000;
	int i;
	i = 0;
	while (i < 4) {
		printf("%d ", a[i]);
		i++;
	}
	printf("\n");
	int b[5] = {2, 20, 200, 2000, 20000};
	i = 0;
	while (i < 5) {
		printf("%d ", b[i]);
		i++;
	}
	printf("\n");
	printf("main() finish!\n");
}
```



Структуры

https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.3

Структура - это объединение одного или нескольких объектов (переменных, массивов, указателей, других структур и т.д.).
Как и массив, она представляет собой совокупность данных. Отличием является то, что к ее элементам необходимо обращаться по имени и что различные элементы структуры не обязательно должны принадлежать одному типу.

Объявление структуры осуществляется с помощью ключевого слова struct, за которым идет ее тип и далее список элементов, заключенных в фигурные скобки:

```
struct тип {
        тип элемента_1 имя элемента_1;
          .........
        тип элемента_n имя элемента_n;
};
```

Структура - пример

```c
#include <stdio.h>

struct data { int d, m, y; };

void main() {
    printf("main() start!\n");

    struct data d1;
    d1.d = 11;
    d1.m = 3;
    d1.y = 2022;

    struct data d2 = { 31, 12, 2021 };

    printf("(%02d.%02d.%d)\n", d1.d, d1.m, d1.y);
    printf("(%02d.%02d.%d)\n", d2.d, d2.m, d2.y);

    printf("main() finish!\n");
}
```

typedef

https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.4

Рассмотрим описание структуры:

```
struct data {int d, m, y;};
```

Здесь фактически вводится новый тип данных - data. Теперь его можно использовать для объявления конкретных экземпляров структуры, например:

```
struct data a, b, c;
```

В язык Си введено специальное средство, позволяющее назначать имена типам данных (переименовывать). Таким средством является оператор typedef. Он записывается в следующем виде:

```
typedef тип имя;
```

Здесь "тип" - любой разрешенный тип данных и "имя" - любой разрешенный идентификатор.

Рассмотрим пример:

```
typedef int INTEGER;
```

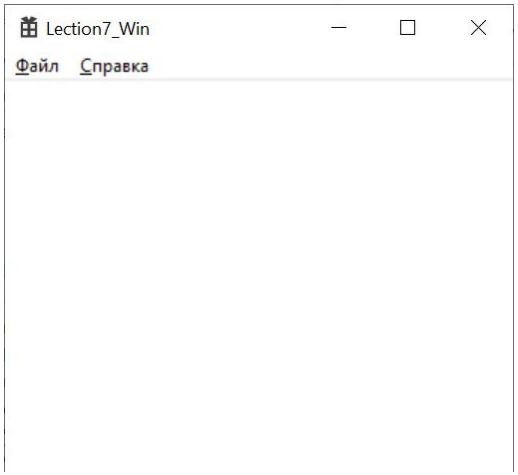
После этого можно сделать объявление:

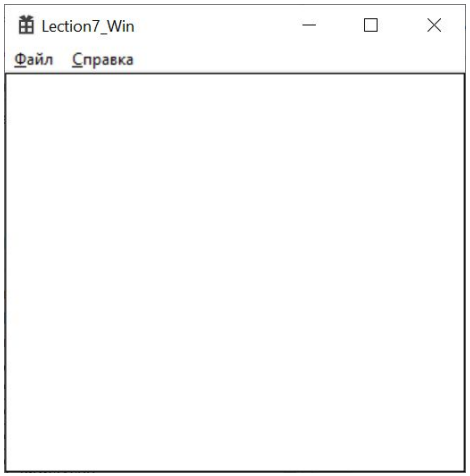
```
INTEGER a, b;
```

Оно будет выполнять то же самое, что и привычное объявление int a,b;. Другими словами, INTEGER можно использовать как синоним ключевого слова int.

struct и typedef – пример (RECT)

```
// windef.h?
typedef struct tagRECT
{
    LONG    left;
    LONG    top;
    LONG    right;
    LONG    bottom;
} RECT;
…
```

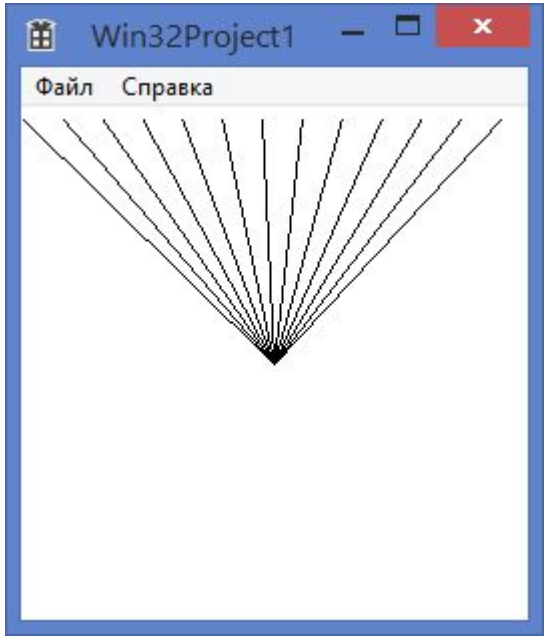





```
// RECT - Структура, в которой хранятся параметры прямоугольника
RECT rect; // struct tagRECT rect;
//Определяем размер клиентской области окна
GetClientRect(hWnd, &rect);
// Рисуем прямоугольник по границам клиентской области окна
Rectangle(hdc, rect.left, rect.top, rect.right, rect.bottom);
```

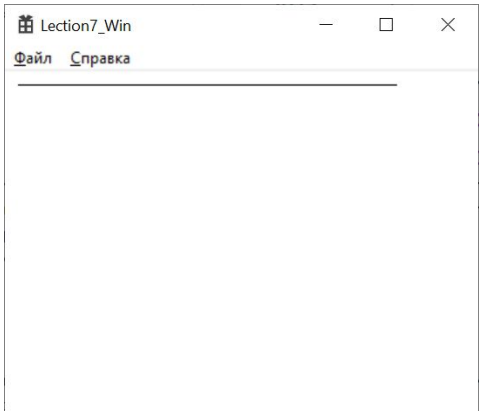
Рисуем много линий из центра

```
case WM_PAINT:
    {
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hWnd, &ps);

    RECT rect;
        GetClientRect(hWnd, &rect);

        int cx = rect.right / 2;
        int cy = rect.bottom / 2;
    int x = 0;
    while (x < rect.right) {
                MoveToEx(hdc, cx, cy, NULL);
                LineTo(hdc, x, 5);
                x += 20;
        }
    EndPaint(hWnd, &ps);
    }
```



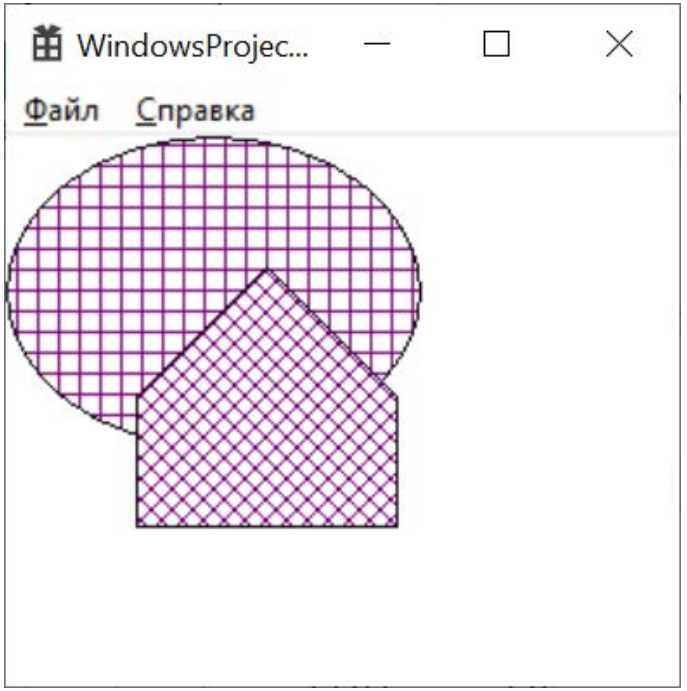
Использование POINT

```
// windef.h?
typedef struct tagPOINT
{
    LONG  x;
    LONG  y;
} POINT;


...

POINT p1;
p1.x = 10;
p1.y = 10;

POINT p2 = { 300, 10 };

MoveToEx(hdc, p1.x, p1.y, NULL);
LineTo(hdc, p2.x, p2.y);
```



Массивы&Структуры – пример - Polygon

```
HBRUSH hBrush = CreateHatchBrush(HS_CROSS, RGB(128, 0, 128));
SelectObject(hdc, hBrush);
Ellipse(hdc, 0, 0 , 160, 120);

HBRUSH hBrush2 = CreateHatchBrush(HS_DIAGCROSS, RGB(128, 0, 128));
SelectObject(hdc, hBrush2);

POINT pt[5];
pt[0].x = 100;
pt[0].y = 50;
pt[1].x = 150;
pt[1].y = 100;
pt[2].x = 150;
pt[2].y = 150;
pt[3].x = 50;
pt[3].y = 150;
pt[4].x = 50;
pt[4].y = 100;
Polygon(hdc, pt, 5);
```

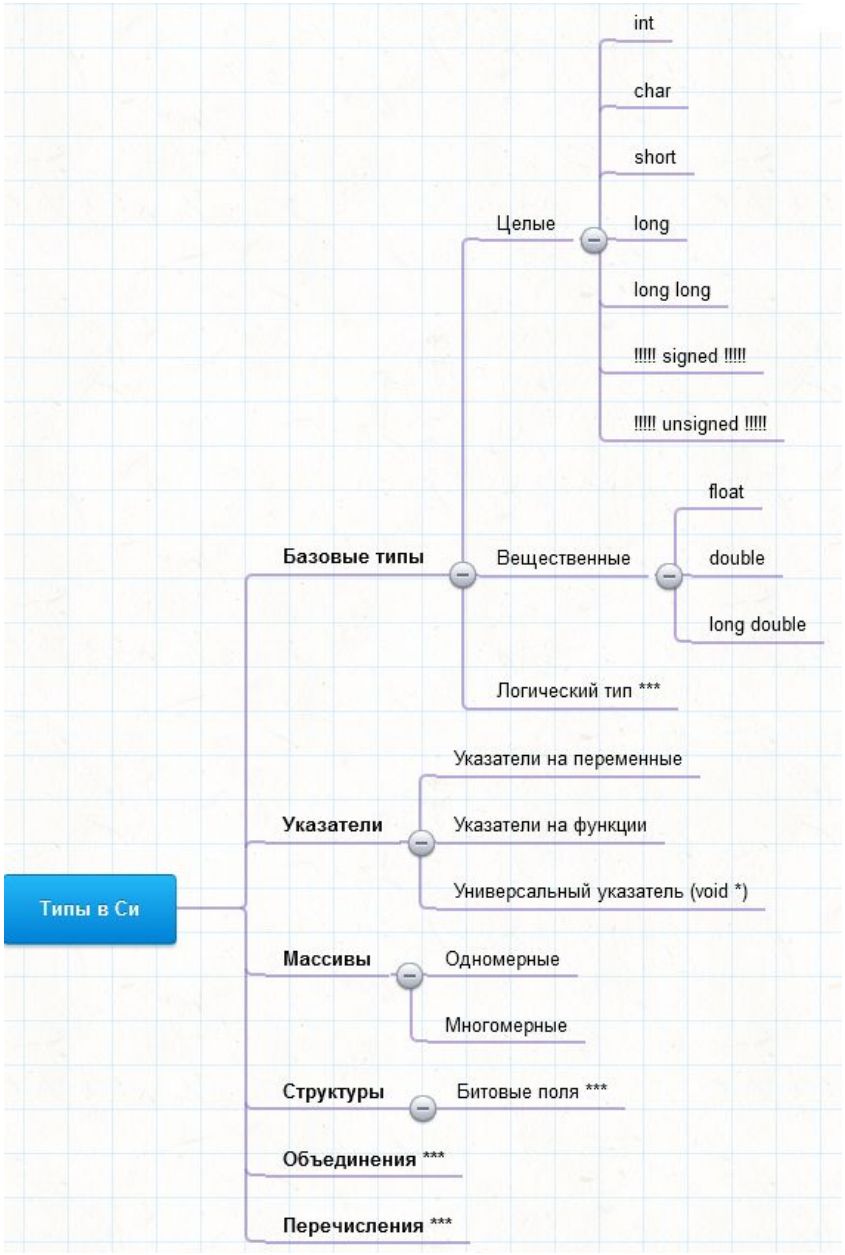


Массивы&Структуры – размещение в памяти

```
POINT pt[5];
pt[0].x = 100;
pt[0].y = 50;
pt[1].x = 150;
pt[1].y = 100;
pt[2].x = 150;
pt[2].y = 150;
pt[3].x = 50;
pt[3].y = 150;
pt[4].x = 50;
pt[4].y = 100;
Polygon(hdc, pt, 5);
```

элемент массива	*имя поля*	*адрес*	*значение*
pt[0]	pt[0].x	**0x0136f6b0**	100
	pt[0].y	**0x0136f6b4**	50
pt[1]	pt[1].x	**0x0136f6b8**	150
	pt[1].y	**0x0136f6bc**	100
pt[2]	pt[2].x	**0x0136f6c0**	150
	pt[2].y	**0x0136f6c4**	150
pt[3]	pt[3].x	**0x0136f6c8**	50
	pt[3].y	**0x0136f6cc**	150
pt[4]	pt[4].x	**0x0136f6d0**	50
	pt[4].y	**0x0136f6d4**	100
		0x0136f6d8	

Какие типы есть в Си?



printf/scanf и базовые типы

Тип	Спецификатор формата
char	%c
signed char	%c (также %d или %hhi (%hhx, %hho) для вывода в числовой форме)
unsigned char	%c (или %hhu для вывода в числовой форме)
short short int signed short signed short int	%hi
unsigned short unsigned short int	%hu
int signed signed int	%i или %d
unsigned unsigned int	%u

Тип	Спецификатор формата
long long int signed long signed long int	%li или %ld
unsigned long unsigned long int	%lu
long long long long int signed long long signed long long int	%lli или %lld
unsigned long long unsigned long long int	%llu
float	%f (автоматически преобразуется в double для printf())
double	%f (%F) (%lf (%lF) для scanf()) %g %G %e %E (для научной нотации)[6]
long double	%Lf %LF %Lg %LG %Le %LE[

https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2_%D0%A1%D0%B8

Где прочитать про типы данных в Си?

С.Ю. Курсков Введение в язык Си

Типы данных - https://dfe.petrsu.ru/koi/posob/c/c.htm#g1.2

Указатели и операции с ними - https://dfe.petrsu.ru/koi/posob/c/c.htm#g2.3

Массивы - https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.1

Структуры - https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.3

Битовые поля - https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.5

Объединение (union) - https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.6

Перечислимый тип данных - https://dfe.petrsu.ru/koi/posob/c/c.htm#g3.7

Указатели на функции - https://dfe.petrsu.ru/koi/posob/c/c.htm#g4.3

***Система типов Си -*
https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2_%D0%A1%D0%B8

*****Б. Керниган, Д. Ритчи Язык программирования Си*

Типы и размеры данных - http://givi.olnd.ru/kr2/02.html#c0202

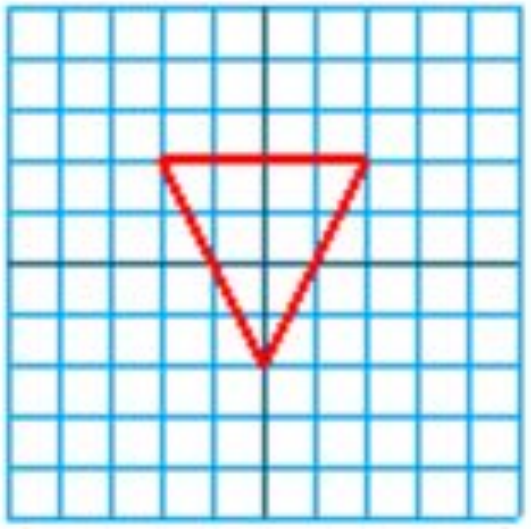
Лабораторная работа №12

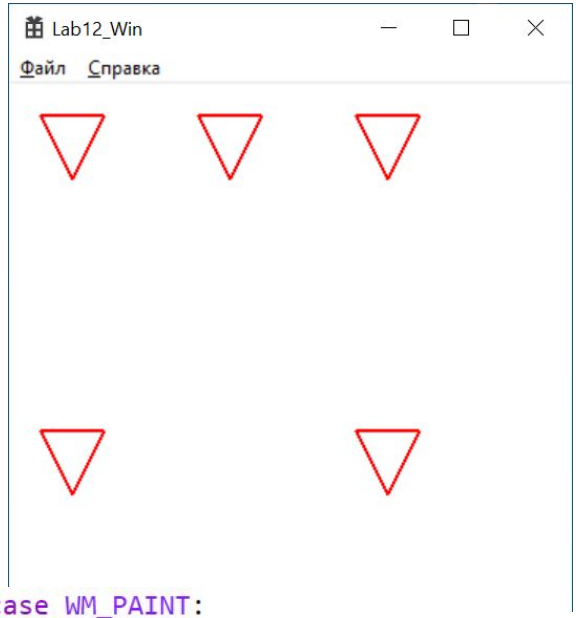
Использование Polyline и Polygon

Задача 1. Отрисовка треугольника 1

Функцию Image0() вставить в ваш код и при помощи нее нарисовать от 3 до 5 треугольников в окне приложения.

```
void Image0(HDC hdc, int cx, int cy) {
    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
    SelectObject(hdc, hPen);

    POINT p[4];
    p[0].x = cx;
    p[0].y = cy + 20;
    p[1].x = cx + 20;
    p[1].y = cy - 20;
    p[2].x = cx - 20;
    p[2].y = cy - 20;
    p[3].x = cx;
    p[3].y = cy + 20;

    Polyline(hdc, p, 4);
    DeleteObject(hPen);
}
```



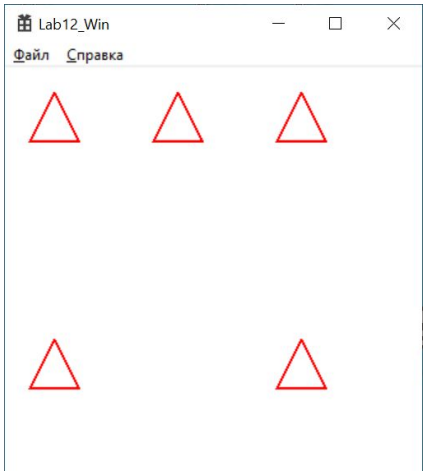



```
case WM_PAINT:
{
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hWnd, &ps);
    Image0(hdc, 40, 40);
    Image0(hdc, 140, 40);
    Image0(hdc, 240, 40);

    Image0(hdc, 240, 240);
    Image0(hdc, 40, 240);

    EndPaint(hWnd, &ps);
}
break;
```

Задача 2. Отрисовка треугольника 2

Функцию Image1() вставить в ваш код и при помощи нее нарисовать от 3 до 5 треугольников в окне приложения.

```
void Image1(HDC hdc, int cx, int cy) {
    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
    SelectObject(hdc, hPen);

    POINT p[4] = {
        cx,        cy - 20,
        cx + 20,  cy + 20,
        cx - 20,  cy + 20,
        cx,        cy - 20
    };
    Polyline(hdc, p, 4);

    DeleteObject(hPen);
}
```

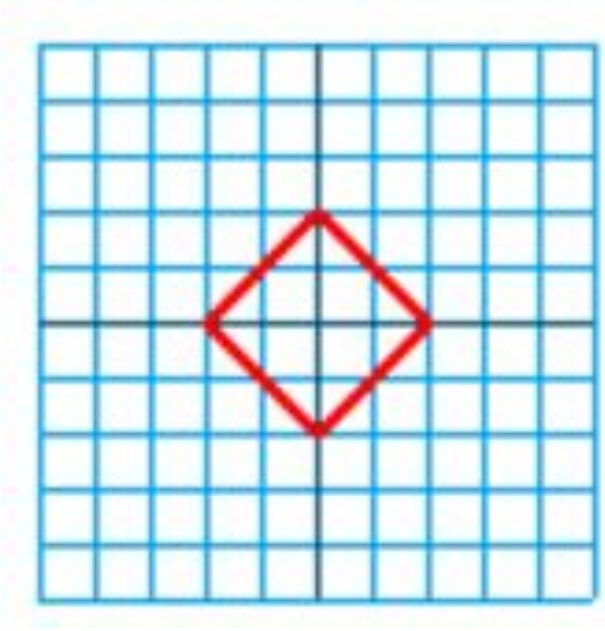


```
case WM_PAINT:
{
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hWnd, &ps);
    Image1(hdc, 40, 40);
    Image1(hdc, 140, 40);
    Image1(hdc, 240, 40);

    Image1(hdc, 240, 240);
    Image1(hdc, 40, 240);

    EndPaint(hWnd, &ps);
}
break;
```

Задача 3. Отрисовка ромба

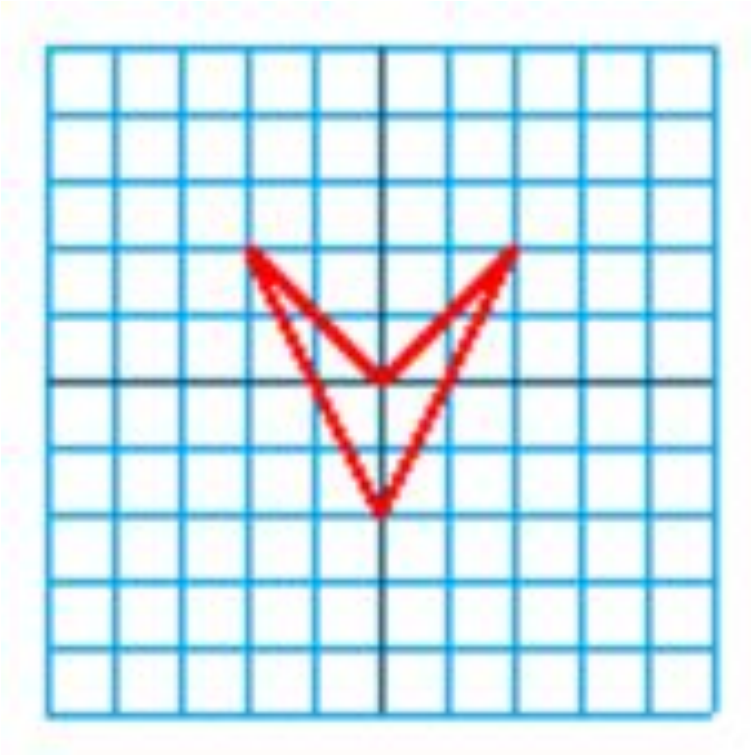
Нужно создать функцию Image2() и при помощи нее нарисовать от 3 до 5 ромбиков в окне приложения.

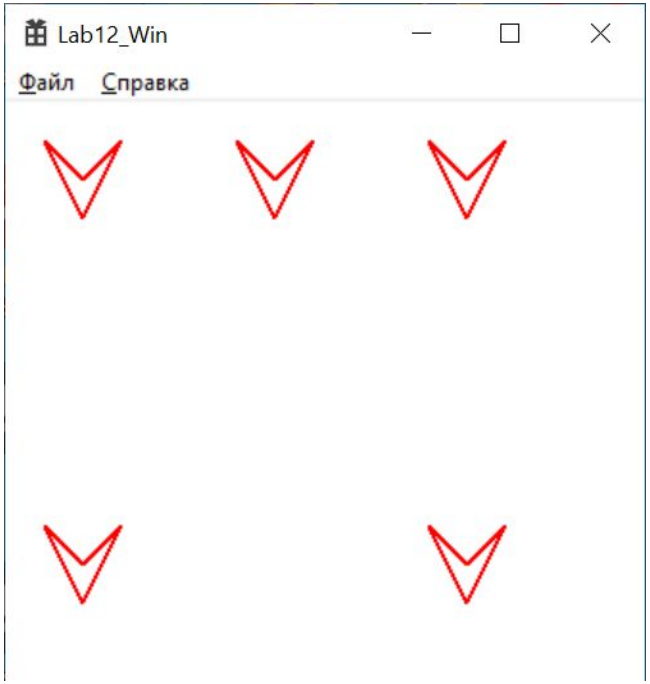
```
void Image2(HDC hdc, int cx, int cy) {

    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
    SelectObject(hdc, hPen);

    POINT p[5] = {
        cx,          cy - 20,
        cx + 20,     cy,
        cx,          cy + 20,
        cx - 20,     cy,
        cx,          cy - 20
    };
    Polyline(hdc, p, 5);

    DeleteObject(hPen);
}
```



Задача 4. Отрисовка сложной фигуры 1

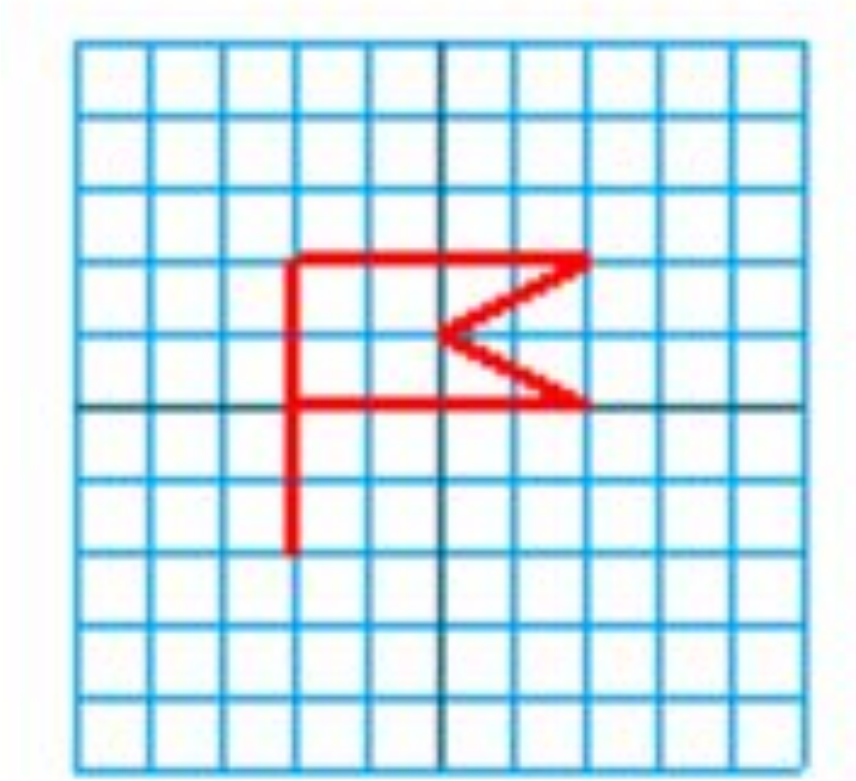
Нужно создать функцию Image3(), в которой отрисовать фигуру по образцу и при помощи неё нарисовать от 3 до 5 фигур в окне приложения.



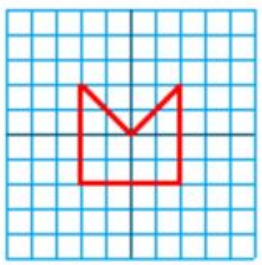
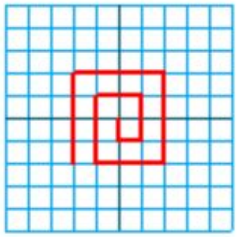
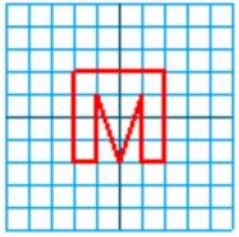


Задача 5*. Отрисовка сложной фигуры 2

Нужно создать функцию Image4(), в которой отрисовать фигуру по образцу и при помощи неё нарисовать от 3 до 5 фигур в окне приложения.



Домашнее задание по ЛР12

1) Доделать задачи 1-4.
2) Сделать функцию Image5() в которой отрисовать следующую фигуру. При помощи Image5() нарисовать от  гур.
3) ** о   е  и 
4) Сделать еще две ваши собственные (уникальные) фигуры.
5) Обязательно! Принести получившийся код на занятие. Его

ИТОГО по ЛР12

1. Научились использовать Polyline/Polygon

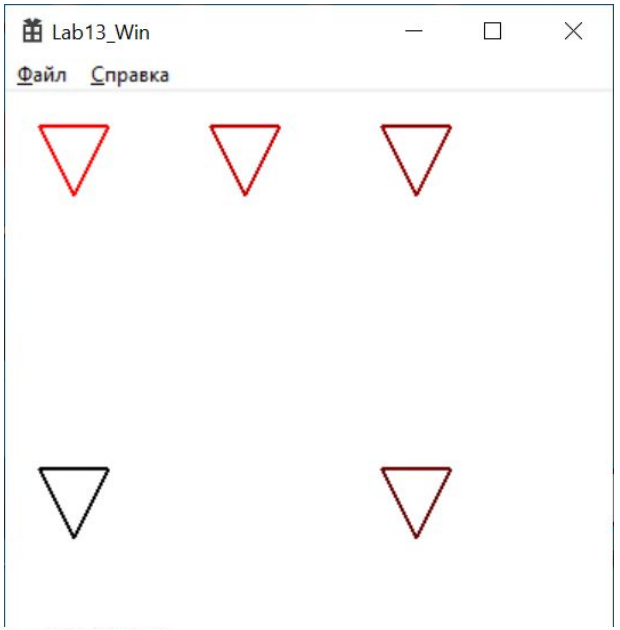
Лабораторная работа №13

Вложенные циклы

Задача 1. Отрисовка треугольника 1 в цвете

Функцию Image0() вставить в ваш код и при помощи нее нарисовать от 3 до 5 треугольников в окне приложения - разным цветом.

```
void Image0(HDC hdc, int cx, int cy, COLORREF color) {
    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, color);
    SelectObject(hdc, hPen);

    POINT p[4] = {
        cx,          cy + 20,
        cx + 20,     cy - 20,
        cx - 20,     cy - 20,
        cx,          cy + 20
    };
    Polyline(hdc, p, 4);

    DeleteObject(hPen);
}
```

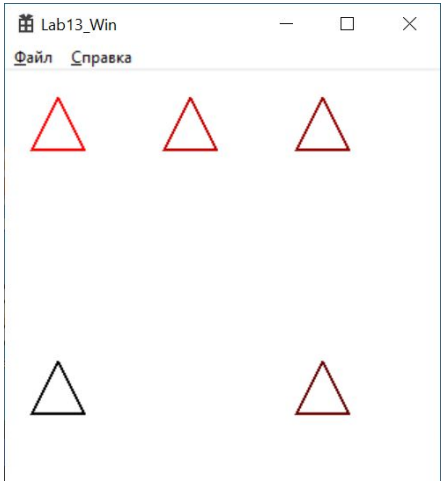



```
case WM_PAINT:
{
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hWnd, &ps);

    Image0(hdc, 40, 40, RGB(255, 0, 0));
    Image0(hdc, 140, 40, RGB(200, 0, 0));
    Image0(hdc, 240, 40, RGB(150, 0, 0));

    Image0(hdc, 240, 240, RGB(100, 0, 0));
    Image0(hdc, 40, 240, RGB(0, 0, 0));
    EndPaint(hWnd, &ps);
}
break;
```

Задача 2. Отрисовка треугольника 2 в цвете

Функцию Image1() из предыдущей лабораторной работы переделайте таким образом, чтобы она могла отрисовывать треугольники разным цветом.

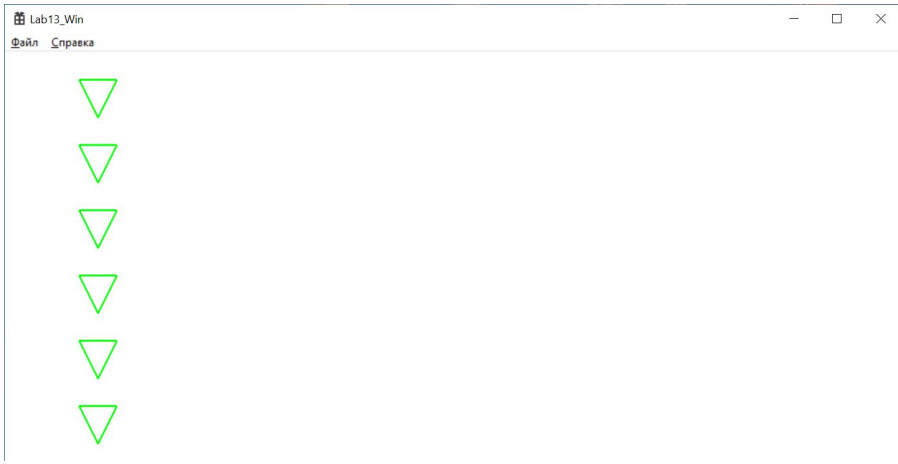
```
void Image1(HDC hdc, int cx, int cy, COLORREF color) {
    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, color);
    SelectObject(hdc, hPen);

    POINT p[4] = {
        cx,          cy - 20,
        cx + 20,     cy + 20,
        cx - 20,     cy + 20,
        cx,          cy - 20
    };
    Polyline(hdc, p, 4);

    DeleteObject(hPen);
}
```



Задача 3. Вертикальный ряд фигур

При помощи функции Image0() нарисовать вертикальный ряд фигур.

```
void PictureV(HDC hdc) {
     int x, y, i;

     x = 100;
     y = 50;
     i = 0;
     do {

          Image0(hdc, x, y, RGB(0, 255, 0));
          y += 70;

          i++;
     } while (i < 6);
}
```

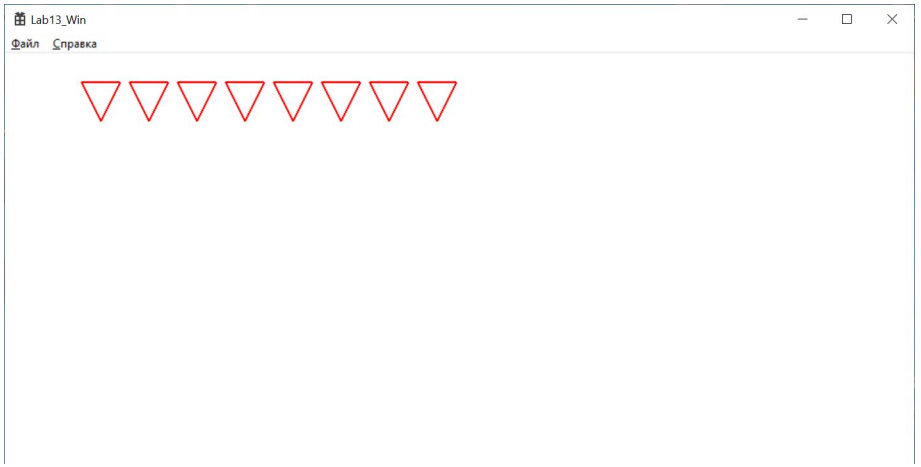



```
case WM_PAINT:
{
     PAINTSTRUCT ps;
     HDC hdc = BeginPaint(hWnd, &ps);
     PictureV(hdc);
     EndPaint(hWnd, &ps);
}
break;
```

Задача 4. Горизонтальный ряд фигур

При помощи функции Image0() нарисовать горизонтальный ряд фигур.

```
void PictureH(HDC hdc) {
     int x, y, j;

     x = 100;
     y = 50;
     j = 0;
     do {
          Image0(hdc, x, y, RGB(255, 0, 0));
          x += 50;

          j++;
     } while (j < 8);
}
```

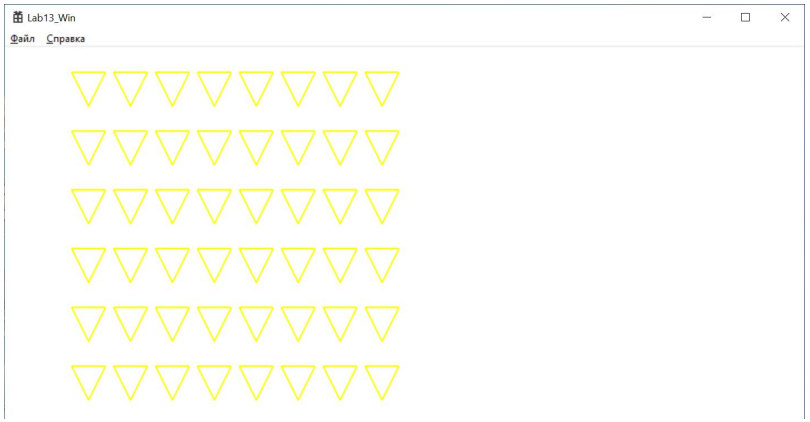



```
case WM_PAINT:
{
     PAINTSTRUCT ps;
     HDC hdc = BeginPaint(hWnd, &ps);
     PictureH(hdc);
     EndPaint(hWnd, &ps);
}
break;
```

Задача 5. Много рядов фигур

При помощи функции Image0() нарисовать много рядов фигур. Использовать для этого вложенные циклы.

```
void PictureVH(HDC hdc) {
     int x, y, i, j;

     y = 50;
     i = 0;
     do {

          x = 100;
          j = 0;
          do {
               Image0(hdc, x, y, RGB(255, 255, 0));
               x += 50;

               j++;
          } while (j < 8);

          y += 70;
          i++;
     } while (i < 6);
}
```




```
case WM_PAINT:
{
     PAINTSTRUCT ps;
     HDC hdc = BeginPaint(hWnd, &ps);
     PictureVH(hdc);
     EndPaint(hWnd, &ps);
}
break;
```

Задача 6*. Изменение цвета 1

При помощи функции Image0() нарисовать вертикальный ряд фигур – с изменением цвета.

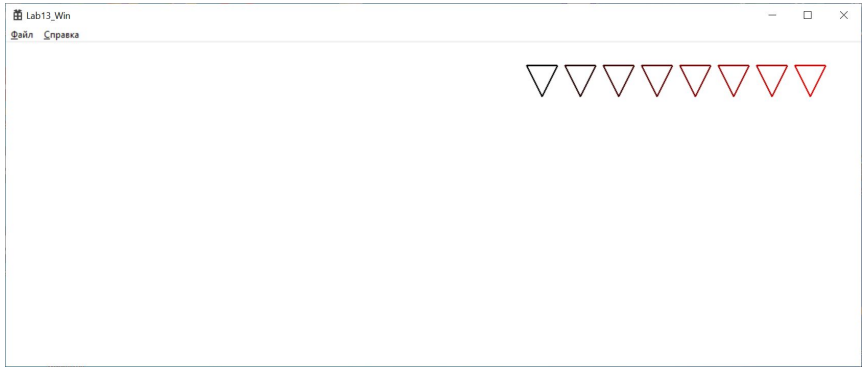
```
void PictureV2(HDC hdc) {
    int x, y, i;
    unsigned char g;

    x = 700;
    y = 50;
    i = 0;
    g = 0;
    do {

        Image0(hdc, x, y, RGB(0, g, 0));
        y += 70;
        g += 40;

        i++;
    } while (i < 6);
}
```


```
case WM_PAINT:
{
     PAINTSTRUCT ps;
     HDC hdc = BeginPaint(hWnd, &ps);
     PictureV2(hdc);
     EndPaint(hWnd, &ps);
}
break;
```

Задача 7*. Изменение цвета 2

При помощи функции Image0() нарисовать вертикальный ряд фигур – с изменением цвета.

```
void PictureH2(HDC hdc) {
    int x, y, j;
    unsigned char r;

    x = 700;
    y = 50;
    j = 0;
    r = 0;
    do {

        Image0(hdc, x, y, RGB(r, 0, 0));
        x += 50;
        r += 30;
        j++;
    } while (j < 8);
}
```




```
case WM_PAINT:
{
     PAINTSTRUCT ps;
     HDC hdc = BeginPaint(hWnd, &ps);
     PictureH2(hdc);
     EndPaint(hWnd, &ps);
}
break;
```

Задача 8*. Изменение цвета 3

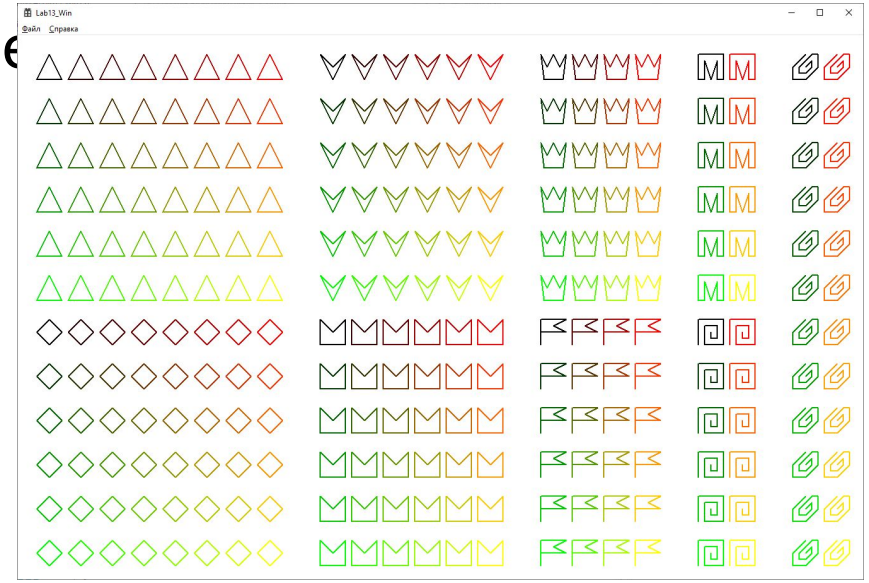
При помощи функции Image0() нарисовать вертикальный ряд фигур – с изменением цвета.

```
void PictureV2H2(HDC hdc) {
    int x, y, i, j;
    unsigned char r, g;
    y = 150;
    i = 0;
    g = 0;
    do {


        x = 800;
        j = 0;
        r = 0;
        do {

            Image0(hdc, x, y, RGB(r, g, 0));
            x += 50;
            r += 36;

            j++;
        } while (j < 8);
        y += 70;
        g += 51;
        i++;
    } while (i < 6);
}
```

```
case WM_PAINT:
{
     PAINTSTRUCT ps;
     HDC hdc = BeginPaint(hWnd, &ps);
     PictureV2H2(hdc);
     EndPaint(hWnd, &ps);
}
break;
```

Домашнее задание по ЛР13

1) Доделать задачи 1-5.
2) * Доделать задачи 6-8
3) Все сделанные в предыдущей лаб работе функции отрисовки фигур Image2(), Image3() и т.д. – переделать таким образом, чтобы можно было рисовать фигуры разным цветом.
4) Используя самые красивые 3 фигуры, созданные вами, создать рисунки из множества рядов из каждой из этих фигур.
5) ** Используя все созданные подобную картину:



ИТОГО по ЛР13

1. Познакомились с вложенными циклами

ИТОГО по лекции 7

1. Узнали практически всё про типы в Си – базовые: целые, вещественные, про указатели, про массивы и структуры.
2. Узнали как при помощи Polyline и Polygon рисовать сложные фигуры.
3. Узнали что нужно сделать в ЛР12 и ЛР13