Курс «Основы программирования»

Власенко Олег Федосович
SimbirSoft

Лекция 11

Строки

ЛР 20. Знакомство с обработкой строк и символов

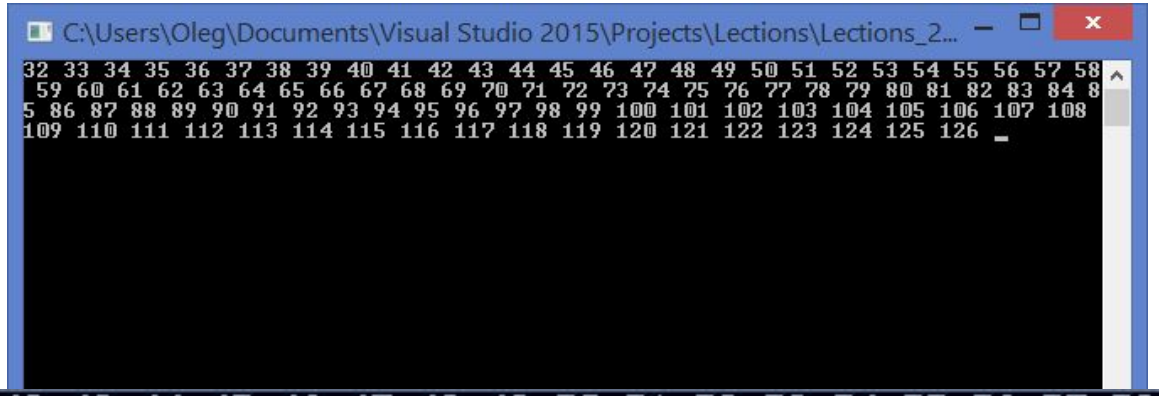
ЛР 21. Изучение стандартной библиотеки Си

Работа с символами в Си: char, ASCII, ASCIIZ, функции isX

Тип char

char – это «очень короткий» целый тип

```
#include <stdio.h>

void main() {

    char ch = 32;
    while (ch < 127) {
        printf("%d ", ch);
        ch++;
    }
}
```

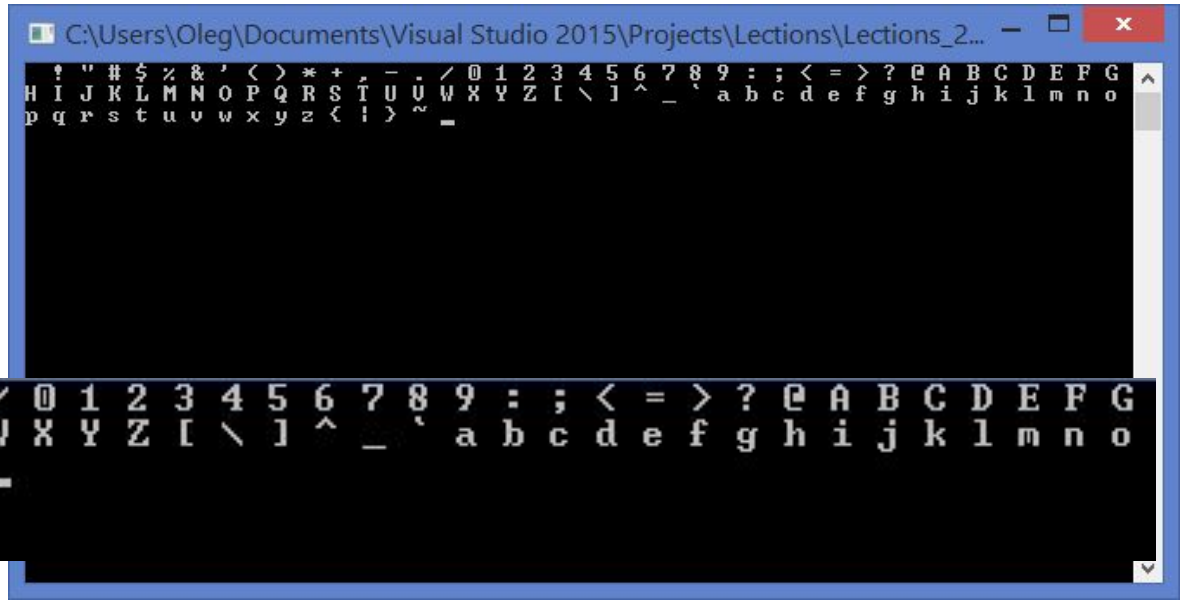


32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 8
5 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108
109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126

Тип char (2)

char – это символьный тип

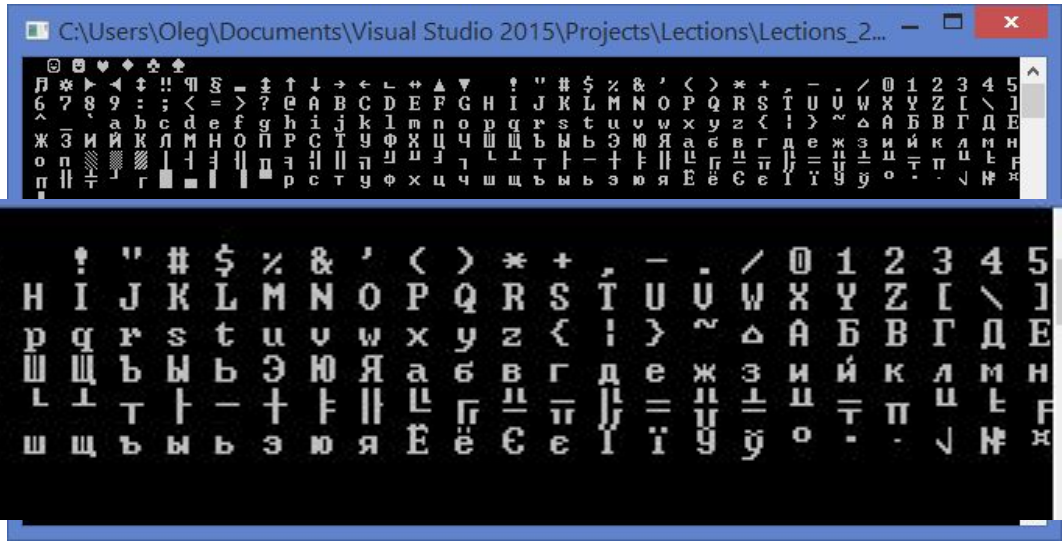
```
#include <stdio.h>

void main() {

    char ch = 32;
    while (ch < 127) {
        printf("%c ", ch);
        ch++;
    }
}
```



Тип char (3)

unsigned char = [0 .. 255]

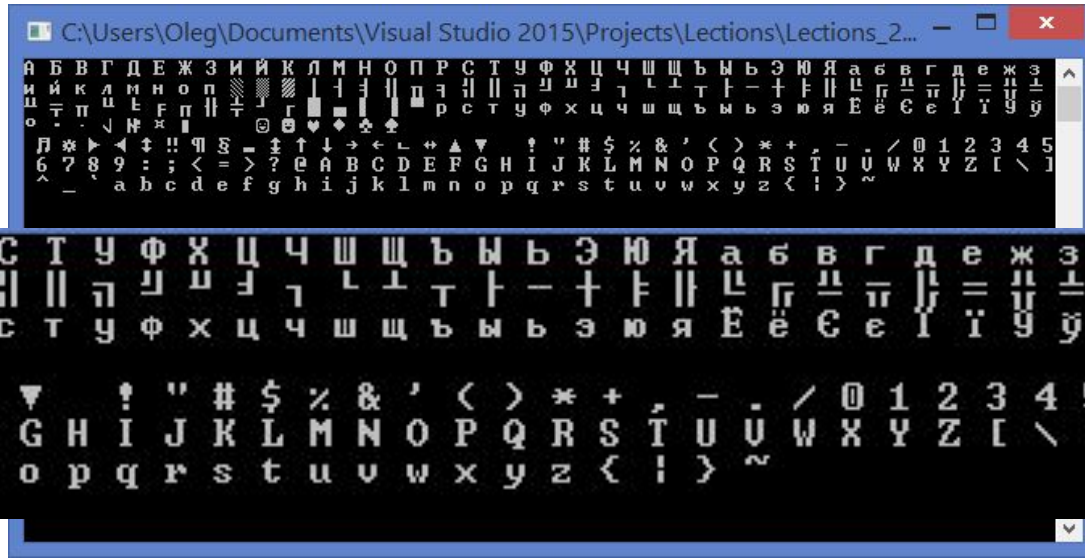
```
#include <stdio.h>

void main() {

    unsigned char ch = 0;

    while (ch < 255) {
        printf("%c ", ch);
        ch++;
    }
}
```



Тип char (4)

signed char = [-128 .. +127]

```
#include <stdio.h>

void main() {

    signed char ch = -128;

    while (ch < 127) {
        printf("%c ", ch);
        ch++;
    }
}
```



Тип char (5)

Загадка:

Тип char == signed char

ИЛИ

Тип char == unsigned char

?

Тип char (6)

http://stackoverflow.com/questions/2054939/is-char-signed-or-unsigned-by-default

The standard does not specify if plain char is signed or unsigned…

ASCII

https://ru.wikipedia.org/wiki/ASCII

ASCII (англ. ***A**merican **s**tandard **c**ode for **i**nformation **i**nterchange*) — название таблицы (кодировки, набора), в которой некоторым распространённым печатным и непечатным символам сопоставлены числовые коды. Таблица была разработана и стандартизована в США в 1963 году.

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Таблица ASCII

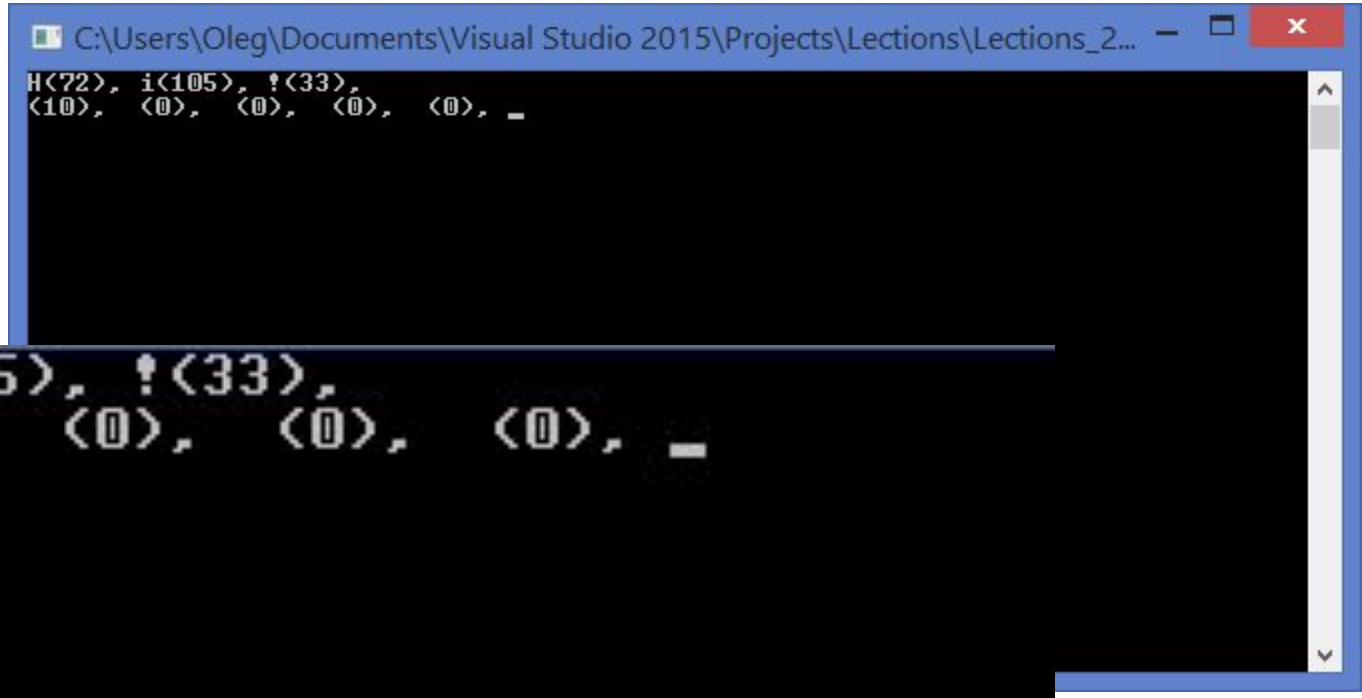
ASCIIZ

http://stackoverflow.com/questions/7783044/whats-the-difference-between-asciiz-vs-ascii

In computing, a C string is a character sequence terminated with a null character ('\0', called NUL in ASCII). It is usually stored as one-dimensional character array.[dubious – discuss] The name refers to the C programming language which uses this string representation. Alternative names are ASCIIZ (note that C strings do not imply the use of ASCII) and **null-terminated string**

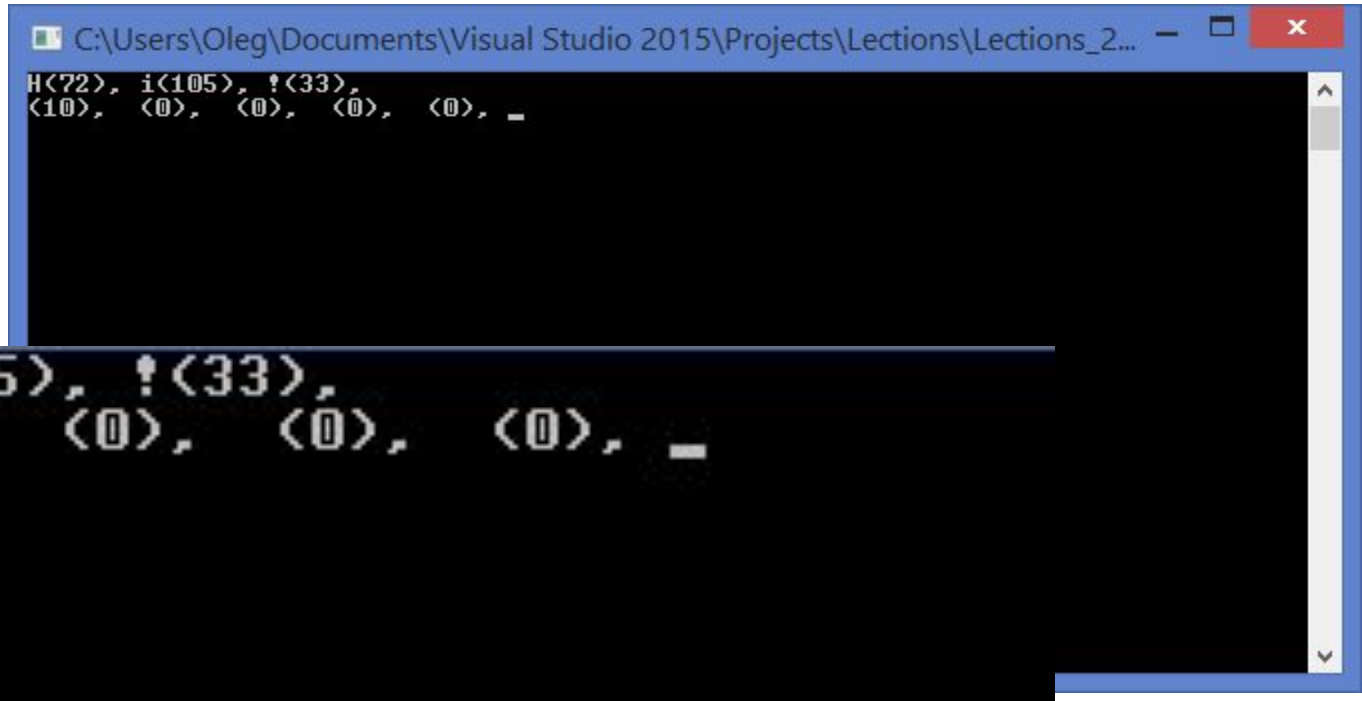
null-terminated string

```
void main() {
	char s1[8] = "Hi!\n";

	int i;
	for (i = 0; i < 8; i++) {
		printf("%c(%d), ", s1[i], s1[i]);
	}
}
```



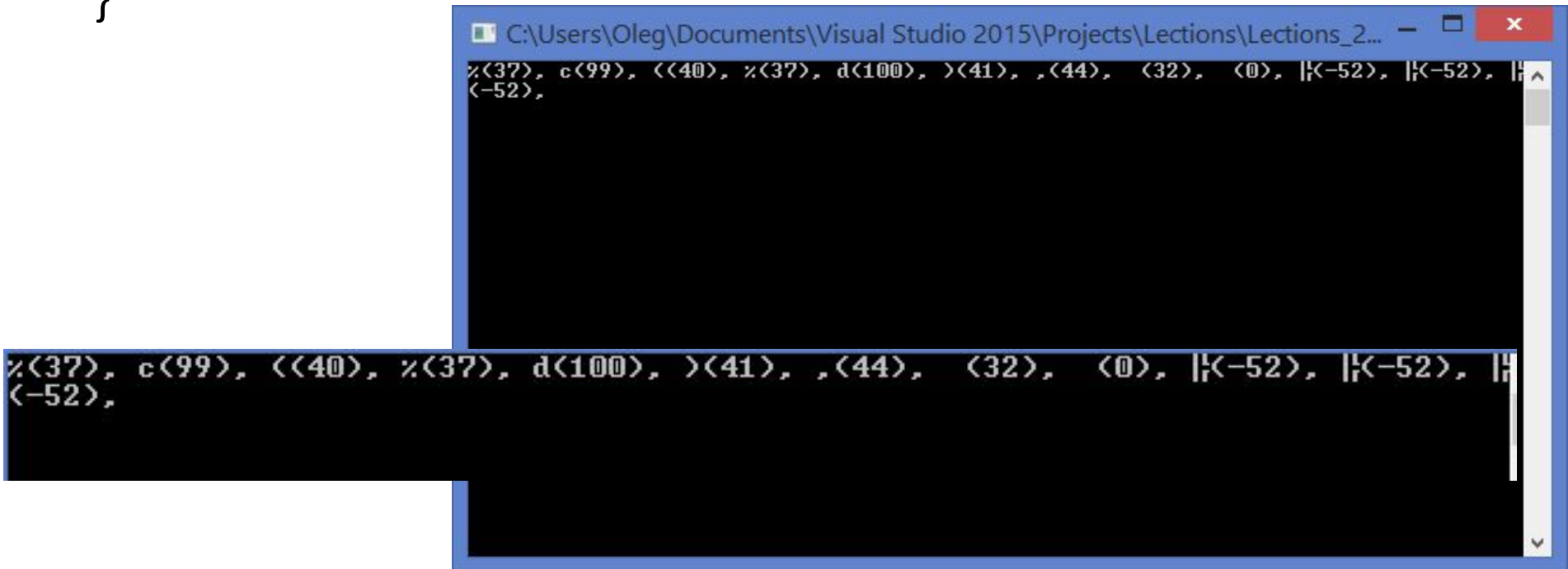
Инициализация строки как массива символов

```
void main() {
	char s1[8] = { 'H', 'i', '!', '\n', '\0' };

	int i;
	for (i = 0; i < 8; i++) {
		printf("%c(%d), ", s1[i], s1[i]);
	}
}
```



Инициализация строки как строки

```
void main() {
	char s2[] = "%c(%d), ";

	int i;
	for (i = 0; i < 12; i++) {
		printf("%c(%d), ", s2[i], s2[i]);
	}
}
```



Простейшие алгоритмы обработки строк (как массива символов с ‘\0’ в конце)

Все цифры заменить на символ «#»

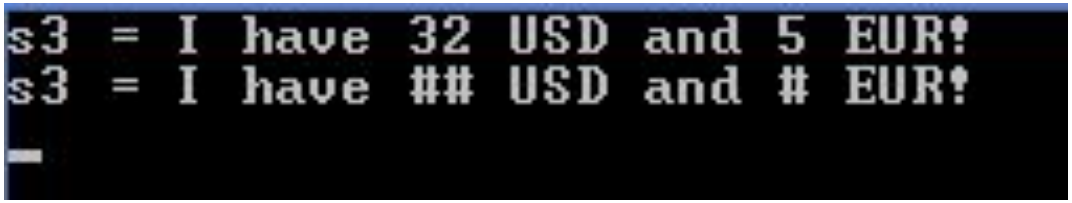
```
#include <stdio.h>
void main() {
    char s3[] = "I have 32 USD and 5 EUR!";
    printf("s3 = %s\n", s3);
    int i = 0;
    while (s3[i] != '\0') {
        if (s3[i] >= '0' && s3[i] <= '9') {
            s3[i] = '#';
        }
        i++;
    }
    printf("s3 = %s\n", s3);
}
```

```
s3 = I have 32 USD and 5 EUR!
s3 = I have ## USD and # EUR!
```

Используем функции из ctype.h

Все цифры заменить на символ «#»

```
#include <stdio.h>
#include <ctype.h>
void main() {
	char s3[] = "I have 32 USD and 5 EUR!";
	printf("s3 = %s\n", s3);
	int i = 0;
	while (s3[i] != '\0') {
		if (isdigit(s3[i])) {
			s3[i] = '#';
		}
		i++;
	}
	printf("s3 = %s\n", s3);
}
```



Используем функции из ctype.h

Все ????? заменить на символ «#»

```
void main() {
    char s3[] = "I have 32 USD and 5 EUR!";
    printf("s3 = %s\n", s3);
    int i = 0;
    while (s3[i] != '\0') {
        if (isalpha(s3[i])) {
            s3[i] = '#';
        }
        i++;
    }
    printf("s3 = %s\n", s3);
}
```

```
s3 = I have 32 USD and 5 EUR!
s3 = # #### 32 ### ### 5 ###!
```

Используем функции из ctype.h

Все ????? заменить на символ «#»

```
void main() {
	char s3[] = "I have 32 USD and 5 EUR!";
	printf("s3 = %s\n", s3);
	int i = 0;
	while (s3[i] != '\0') {
		if (isspace(s3[i])) {
			s3[i] = '#';
		}
		i++;
	}
	printf("s3 = %s\n", s3);
}
```

```
s3 = I have 32 USD and 5 EUR!
s3 = I#have#32#USD#and#5#EUR!
```

Используем функции из ctype.h

Все ????? заменить на символ «#»

```
void main() {
	char s3[] = "I have 32 USD and 5 EUR!";
	printf("s3 = %s\n", s3);
	int i = 0;
	while (s3[i] != '\0') {
		if (isupper(s3[i])) {
			s3[i] = '#';
		}
		i++;
	}
	printf("s3 = %s\n", s3);
}
```

```
s3 = I have 32 USD and 5 EUR!
s3 = # have 32 ### and 5 ###!
```

Используем функции из ctype.h

Все ????? заменить на символ «#»

```
void main() {
	char s3[] = "I have 32 USD and 5 EUR!";
	printf("s3 = %s\n", s3);
	int i = 0;
	while (s3[i] != '\0') {
		if (islower(s3[i])) {
			s3[i] = '#';
		}
		i++;
	}
	printf("s3 = %s\n", s3);
}
```

```
s3 = I have 32 USD and 5 EUR!
s3 = I #### 32 USD ### 5 EUR!
```

Используем функции из ctype.h

Все ????? заменить на ??????

```
void main() {
    char s3[] = "I have 32 USD and 5 EUR!";

    printf("s3 = %s\n", s3);
    int i = 0;

    while (s3[i] != '\0') {
        s3[i] = toupper(s3[i]);
        i++;
    }

    printf("s3 = %s\n", s3);
}
```

```
s3 = I have 32 USD and 5 EUR!
s3 = I HAVE 32 USD AND 5 EUR!
```

Используем функции из ctype.h

Все ????? заменить на ??????

```
void main() {
    char s3[] = "I have 32 USD and 5 EUR!";

    printf("s3 = %s\n", s3);
    int i = 0;

    while (s3[i] != '\0') {
        s3[i] = tolower(s3[i]);
        i++;
    }

    printf("s3 = %s\n", s3);
}
```

```
s3 = I have 32 USD and 5 EUR!
s3 = i have 32 usd and 5 eur!
```

Обзор возможностей функций из ctype.h

«Основы программирования на языках Си и C++ для начинающих»
Заголовочный файл cctype (ctype.h) - http://cppstudio.com/cat/309/313/

isalumn	Функция возвращает истинное значение true, если её аргумент - буква или цифра, и false(ложь) в других случаях.
isalpha	Функция возвращает истинное значение true, если её аргумент - буква, и false(ложь) в других случаях.
iscntrl	Функция возвращает истинное значение true, если её аргумент - управляемый символ, и false(ложь) в других случаях.
isdigit	Функция возвращает истинное значение true, если её аргумент - десятичная цифра, и false(ложь) в других случаях.
isgraph	Функция возвращает истинное значение true, если её аргумент - символ, имеющий графическое представление, и false(ложь) в других случаях.
islower	Функция возвращает истинное значение true, если её аргумент - строчный символ алфавита, и false(ложь) в других случаях.
isprint	Функция возвращает истинное значение true, если её аргумент - печатный символ, и false(ложь) в других случаях.
ispunct	Функция возвращает истинное значение true, если её аргумент - знак пунктуации, и false(ложь) в других случаях.
isspace	Функция возвращает истинное значение true, если её аргумент - любой знак пробела, и false(ложь) в других случаях.
isupper	Функция возвращает истинное значение true, если её аргумент - прописная буква алфавита, и false(ложь) в других случаях.
isxdigit	Функция возвращает истинное значение true, если её аргумент - цифра шестнадцатеричной системы исчисления, и false(ложь) в других случаях.

Обзор возможностей функций из ctype.h (2)

ASCII	символы	isalumn	isalpha	iscntrl	isdigit	isgraph	islower	isprint	isxdigit	ispunct	isspace	isupper
0x00-0x08	NUL и другие управляющие символы			■								
0x09-0x0D	пробелы: 'v', 'n', 't', 'r', 'f'			■							■	
0x0E-0x1F	другие управляющие символы			■								
0x20	пробел ' '							■			■	
0x21-0x2F	! " # $ % & ' ( ) * + , - . /					■		■		■		
0x30-0x39	0 1 2 3 4 5 6 7 8 9	■			■	■		■	■			
0x3A-0x40	: ; < = > ? @					■		■		■		
0x41-0x46	A B C D E F	■	■			■		■	■			■
0x47-0x5A	G H I J K L M N O P Q R S T U V W X Y Z	■	■			■		■				■
0x5B-0x60	[ ] ^ _ `					■		■		■		
0x61-0x66	a b c d e f	■	■			■	■	■	■			
0x67-0x7A	g h i j k l m n o p q r s t u v w x y z	■	■			■	■	■				
0x7B-0x7E	{ | } ~					■		■		■		
0x7F	DEL											

Функции обработки строк

Стандартные функции обработки строк

strlen(s) - Возвращает длину строки без завершающей литеры '\0'.

strcmp(s1, s2) – посимвольное сравнение строк (НЕЛЬЗЯ сравнивать строки так «s1 == s2» или так
«s1 < s2»!!!)

strcpy (dest, src) – копирует сроку src в dest, включая завершающий ‘\0’

strcat (dest, src) – добавляет копию src в конец dest

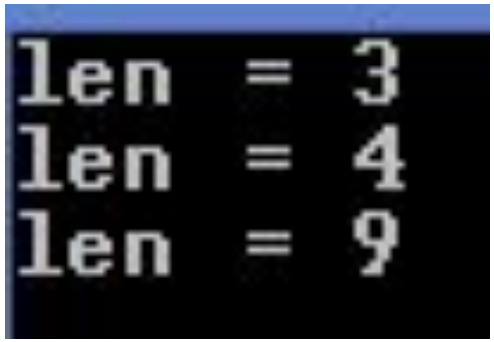
И еще около 20 функций из string.h

strlen()

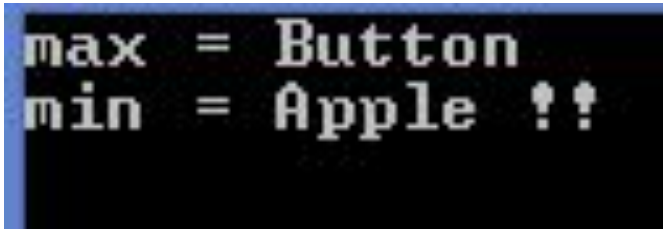
```
#include <string.h>
void main() {
    char s[10] = "Hi!";
    printf("len = %d\n", strlen(s));

    s[3] = ' '; s[4] = '\0';
    printf("len = %d\n", strlen(s));

    s[4] = 'W'; s[5] = 'o'; s[6] = 'r'; s[7] = 'l';
    s[8] = 'd'; s[9] = '\0';
    printf("len = %d\n", strlen(s));
}
```

strlen()

```
#include <string.h>
void main() {
    char s[10] = "Hi!";
    printf("len = %d\n", strlen(s));

    s[3] = ' '; s[4] = '\0';
    printf("len = %d\n", strlen(s));

    s[4] = 'W'; s[5] = 'o'; s[6] = 'r'; s[7] = 'l';
    s[8] = 'd'; s[9] = '\0';
    printf("len = %d\n", strlen(s));
}
```



Сравнение строк – НЕ ДЕЛАЙТЕ ТАК НИКОГДА!!!




```
void main() {
    char s1[] = "Button";
    char s2[] = "We";
    char s3[] = "Apple !!";
    char * min = s1;char * max = s1;

    if (s2 > max) max = s2;
    if (s3 > max) max = s3;
    printf("max = %s\n", max);

    if (s2 < min)  min = s2;
    if (s3 < min)  min = s3;
    printf("min = %s\n", min);
}
```

Сравнение строк через strcmp

```
int strcmp(const char *str1, const char *str2);
int strcmp(char str1[], char str2[]);
```

Функция strcmp() сравнивает в лексикографическом порядке две строки и возвращает целое значение, зависящее следующим образом от результата сравнения.

Значение	*Результат сравнения строк*
Меньше нуля	str1 меньше str2
Нуль	str1 равен str2
Больше нуля	str1 больше str2

Сравнение строк через strcmp

```
max = We
min = Apple !!
```

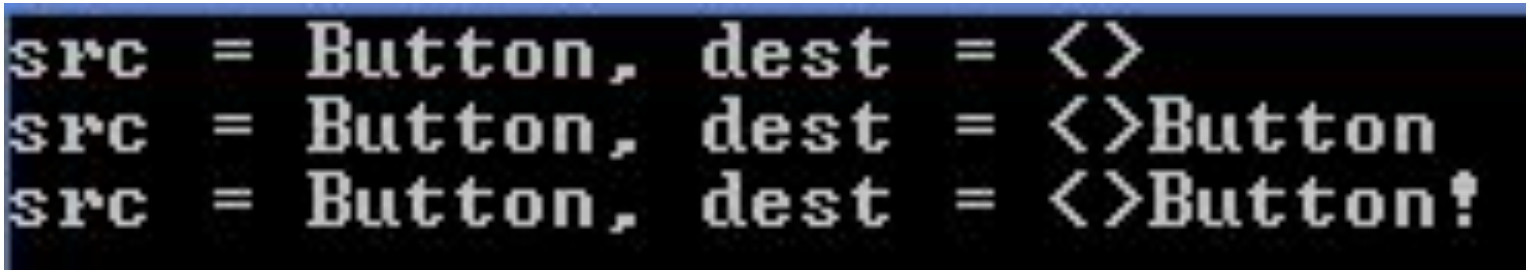
```
void main() {
    char s1[] = "Button";
    char s2[] = "We";
    char s3[] = "Apple !!";
    char * min = s1;char * max = s1;

    if (strcmp(s2, max) > 0) max = s2;
    if (strcmp(s2, max) > 0) max = s3;
    printf("max = %s\n", max);

    if (strcmp(s2, min) < 0) min = s2;
    if (strcmp(s3, min) < 0) min = s3;
    printf("min = %s\n", min);
}
```

Копирование строк

```
void main() {
    char src[] = "Button";
    char dest[10];

    printf("src = %s, dest = %s\n", src, dest);

    strcpy(dest, src);
    printf("src = %s, dest = %s\n", src, dest);

}
```

Конкатенация строк

```
void main() {
    char src[] = "Button";
    char dest[10] = "<>";
    printf("src = %s, dest = %s\n", src, dest);

    strcat(dest, src);
    printf("src = %s, dest = %s\n", src, dest);

    strcat(dest, "!");
    printf("src = %s, dest = %s\n", src, dest);
}
```



Еще раз - int strlen(char s[])

```
int strlen(char s[]) {
    int len;
    …
    return len;
}
```

Возвращает длину строки без завершающей литеры '\0'.

Пример:

strlen(“!!”) == 2

strlen(“Hi!\n”) ==4

Собственная реализация strlen

```
int strlen_my(char s[]) {
    int len;
    …
    return len;
}
```

Нужно написать код функции strlen_my(s), работающей аналогично strlen(s)

Пример использования:

strlen_my(“!!”) == 2

strlen_my(“Hi!\n”) ==4

Собственная реализация strlen

```
int strlen_my(char s[])
{
    int len = 0;

    while (s[len] != '\0')
        len++;

    return len;
}
```

int strcmp(char s1[], char s2[])

int strcmp(const char *str1, const char *str2);

Функция strcmp() сравнивает в лексикографическом порядке две строки и возвращает целое значение, зависящее следующим образом от результата сравнения.

Значение	*Результат сравнения строк*
Меньше нуля	str1 меньше str2
Нуль	str1 равен str2
Больше нуля	str1 больше str2

Пример использования:

strcmp(“Abba”, “Beta”) < 0

Собственная реализация strcmp

```
/*
s1 < s2   : <0
s1 == s2  : 0
s1 > s2   : >0
*/
int strcmp_my(char  s1[], char  s2[])
{
    return …;
}
```

Нужно написать код функции strcmp_my(s1, s2), работающей аналогично strcmp(s1, s2)

Пример использования:

strcmp_my(“Abba”, “Beta”) < 0

Собственная реализация strcmp

```
/*
s1 < s2   : <0
s1 == s2  : 0
s1 > s2   : >0
*/
int strcmp_my(char  s1[], char  s2[])
{
    int i = 0;
    while (s1[i] != 0 && s2[i] != 0 && s1[i] == s2[i])
        i++;

    return s1[i] - s2[i];
}
```

Собственная реализация strcpy

!!!!

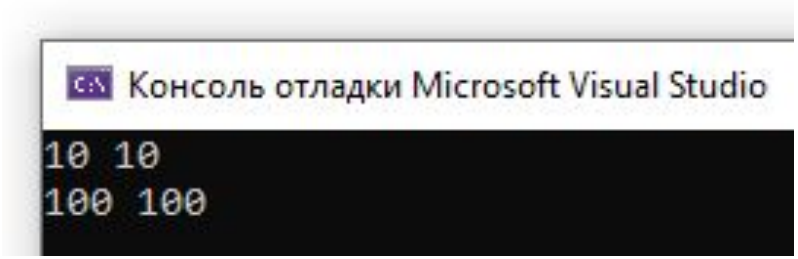
Собственная реализация strcat

!!!!

Массивы и указатели

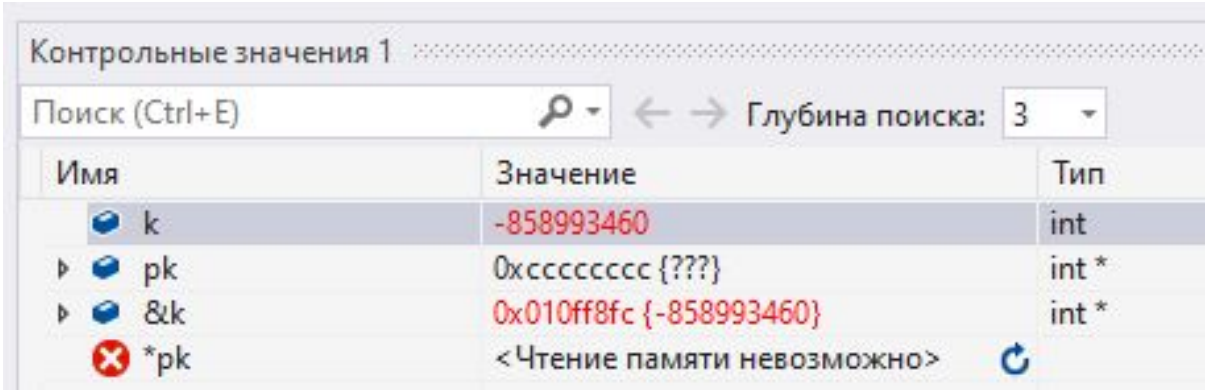
Указатели

```c
#include <stdio.h>

void main() {
    int k;    // переменная k (целый тип)
    int* pk;  // указатель на int

    k = 10;   // переменная k = 10
    pk = &k;  // указатель pk указывает на переменную k
           // *pk - обращение через указатель к переменной
    printf("%d %d\n", k, *pk);

    *pk = 100;
    printf("%d %d\n", k, *pk);
}
```



Указатели – трассировка (1)

```
14  #include <stdio.h>
15
16  void main() {
17      int k;    // переменная k (целый тип)
18      int* pk;  // указатель на int
19
20      k = 10;   // переменная k = 10
21      pk = &k;  // указатель pk указывает на переменную k
22            // *pk – обращение через указатель к переменной
23      printf("%d %d\n", k, *pk);
24
25      *pk = 100;
26      printf("%d %d\n", k, *pk);
27  }
28
```



Указатели – трассировка (2)

```
14  #include <stdio.h>
15
16  void main() {
17      int k;     // переменная k (целый тип)
18      int* pk;   // указатель на int
19
20      k = 10;    // переменная k = 10
21      pk = &k;   // указатель pk указывает на переменную k    ≤ 1 мс прошло
22          // *pk – обращение через указатель к переменной
23      printf("%d %d\n", k, *pk);
24
25      *pk = 100;
26      printf("%d %d\n", k, *pk);
27  }
28
```

Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

Имя	Значение	Тип
k	10	int
pk	0xcccccccc {???}	int *
&k	0x010ff8fc {10}	int *
*pk	<Чтение памяти невозможно>	

Указатели – трассировка (3)

```
14  #include <stdio.h>
15
16  void main() {
17      int k;    // переменная k (целый тип)
18      int* pk;  // указатель на int
19
20      k = 10;   // переменная k = 10
21      pk = &k;  // указатель pk указывает на переменную k
22            // *pk – обращение через указатель к переменной
23      printf("%d %d\n", k, *pk);  ≤ 1 мс прошло
24
25      *pk = 100;
26      printf("%d %d\n", k, *pk);
27  }
28
```

Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

Имя	Значение	Тип
k	10	int
pk	0x010ff8fc {10}	int *
&k	0x010ff8fc {10}	int *
*pk	10	int

Указатели – трассировка (4)

```
14  #include <stdio.h>
15
16  void main() {
17      int k;    // переменная k (целый тип)
18      int* pk;  // указатель на int
19
20      k = 10;   // переменная k = 10
21      pk = &k;  // указатель pk указывает на переменную k
22            // *pk – обращение через указатель к переменной
23      printf("%d %d\n", k, *pk);
24
25      *pk = 100;
26      printf("%d %d\n", k, *pk);   ≤ 1 мс прошло
27  }
28
```

Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

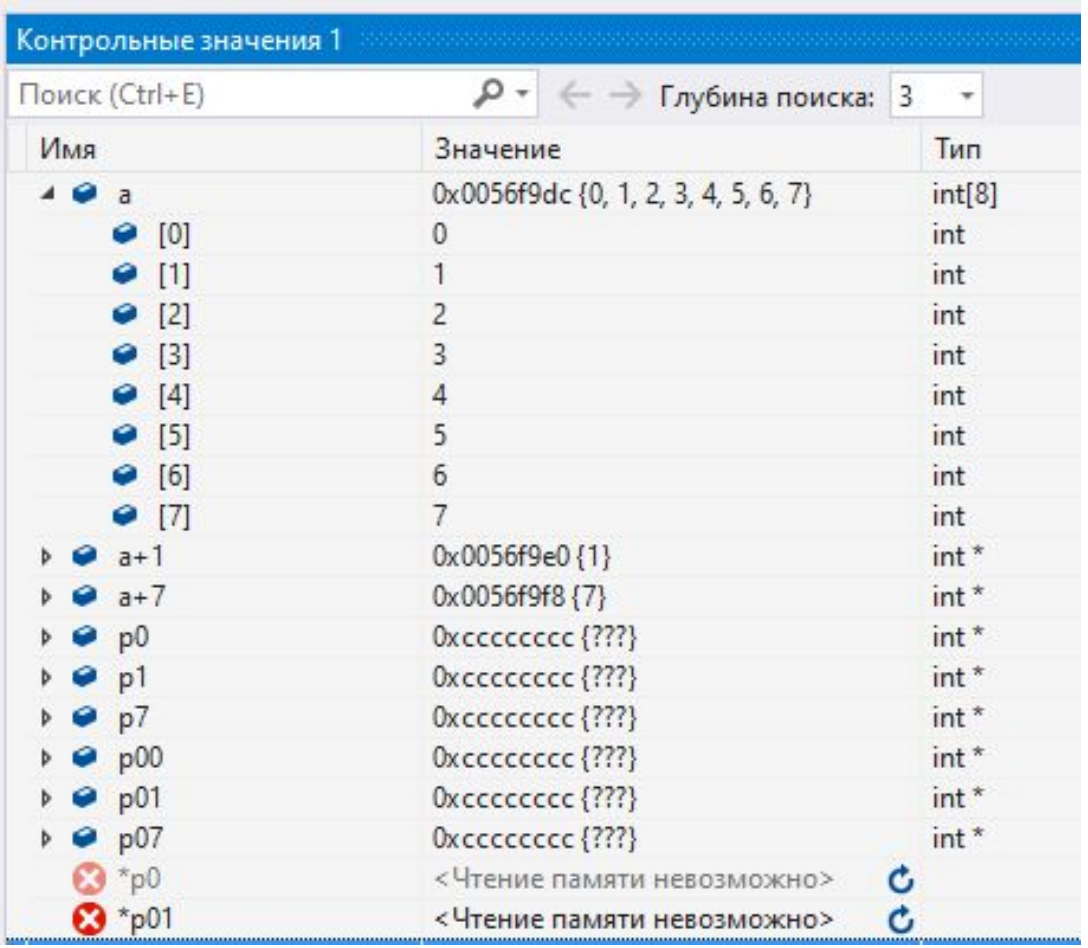
Имя	Значение	Тип
k	100	int
pk	0x010ff8fc {100}	int *
&k	0x010ff8fc {100}	int *
*pk	100	int

Связь массивов и указателей

```
#include <stdio.h>

void main() {
    int a[8] = {0, 1, 2, 3, 4, 5, 6, 7};
    int* p0 = &a[0];
    int* p1 = &a[1];
    int* p7 = &a[7];

    int* p00 = a;
    int* p01 = a + 1;
    int* p07 = a + 7;

    *p0 = 100;
    *p01 = 200;
}
```

Связь массивов и указателей (1)

```
32  #include <stdio.h>
33
34  void main() {
35      int a[8] = {0, 1, 2, 3, 4, 5, 6, 7}; // массив a содержащий 8 элементов целого типа
36
37      int* p0 = &a[0];
38      int* p1 = &a[1];
39      int* p7 = &a[7];
40
41      int* p00 = a;
42      int* p01 = a + 1;
43      int* p07 = a + 7;
44
45      *p0 = 100;
46      *p01 = 200;
47  }
48
```



Связь массивов и указателей (2)

```
34  void main() {
35      int a[8] = {0, 1, 2, 3, 4, 5, 6, 7}; // массив a содержащий 8 элементов целого типа
36
37      int* p0 = &a[0];
38      int* p1 = &a[1];  ≤ 1 мс прошло
39      int* p7 = &a[7];
40
41      int* p00 = a;
42      int* p01 = a + 1;
43      int* p07 = a + 7;
44
45      *p0 = 100;
46      *p01 = 200;
47  }
48
```

Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

Имя	Значение	Тип
a	0x0056f9dc {0, 1, 2, 3, 4, 5, 6, 7}	int[8]
[0]	0	int
[1]	1	int
[2]	2	int
[3]	3	int
[4]	4	int
[5]	5	int
[6]	6	int
[7]	7	int
a+1	0x0056f9e0 {1}	int *
a+7	0x0056f9f8 {7}	int *
p0	0x0056f9dc {0}	int *
p1	0xcccccccc {???}	int *
p7	0xcccccccc {???}	int *
p00	0xcccccccc {???}	int *
p01	0xcccccccc {???}	int *
p07	0xcccccccc {???}	int *
*p0	0	int
*p01	<Чтение памяти невозможно>	

Связь массивов и указателей (3)

```
32  #include <stdio.h>
33
34  void main() {
35      int a[8] = {0, 1, 2, 3, 4, 5, 6, 7}; // массив a содержащий 8 элементов целого типа
36
37      int* p0 = &a[0];
38      int* p1 = &a[1];
39      int* p7 = &a[7];  ≤ 1 мс прошло
40
41      int* p00 = a;
42      int* p01 = a + 1;
43      int* p07 = a + 7;
44
45      *p0 = 100;
46      *p01 = 200;
47  }
48
```

Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

Имя	Значение	Тип
a	0x0056f9dc {0, 1, 2, 3, 4, 5, 6, 7}	int[8]
[0]	0	int
[1]	1	int
[2]	2	int
[3]	3	int
[4]	4	int
[5]	5	int
[6]	6	int
[7]	7	int
a+1	0x0056f9e0 {1}	int *
a+7	0x0056f9f8 {7}	int *
p0	0x0056f9dc {0}	int *
p1	0x0056f9e0 {1}	int *
p7	0xcccccccc {???}	int *
p00	0xcccccccc {???}	int *
p01	0xcccccccc {???}	int *
p07	0xcccccccc {???}	int *
*p0	0	int
*p01	<Чтение памяти невозможно>	

Связь массивов и указателей (4)

```
32  #include <stdio.h>
33
34  void main() {
35      int a[8] = {0, 1, 2, 3, 4, 5, 6, 7}; // массив a содержащий 8 элементов целого типа
36
37      int* p0 = &a[0];
38      int* p1 = &a[1];
39      int* p7 = &a[7];
40
41      int* p00 = a;  ≤ 1 мс прошло
42      int* p01 = a + 1;
43      int* p07 = a + 7;
44
45      *p0 = 100;
46      *p01 = 200;
47  }
48
```

Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

Имя	Значение	Тип
a	0x0056f9dc {0, 1, 2, 3, 4, 5, 6, 7}	int[8]
[0]	0	int
[1]	1	int
[2]	2	int
[3]	3	int
[4]	4	int
[5]	5	int
[6]	6	int
[7]	7	int
a+1	0x0056f9e0 {1}	int *
a+7	0x0056f9f8 {7}	int *
p0	0x0056f9dc {0}	int *
p1	0x0056f9e0 {1}	int *
p7	0x0056f9f8 {7}	int *
p00	0xcccccccc {???}	int *
p01	0xcccccccc {???}	int *
p07	0xcccccccc {???}	int *
*p0	0	int
*p01	<Чтение памяти невозможно>	

Связь массивов и указателей (5)

```
32  #include <stdio.h>
33
34  void main() {
35      int a[8] = {0, 1, 2, 3, 4, 5, 6, 7}; // массив a содержащий 8 элементов целого типа
36
37      int* p0 = &a[0];
38      int* p1 = &a[1];
39      int* p7 = &a[7];
40
41      int* p00 = a;
42      int* p01 = a + 1;  ≤ 1 мс прошло
43      int* p07 = a + 7;
44
45      *p0 = 100;
46      *p01 = 200;
47  }
48
```

Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

Имя	Значение	Тип
a	0x0056f9dc {0, 1, 2, 3, 4, 5, 6, 7}	int[8]
[0]	0	int
[1]	1	int
[2]	2	int
[3]	3	int
[4]	4	int
[5]	5	int
[6]	6	int
[7]	7	int
a+1	0x0056f9e0 {1}	int *
a+7	0x0056f9f8 {7}	int *
p0	0x0056f9dc {0}	int *
p1	0x0056f9e0 {1}	int *
p7	0x0056f9f8 {7}	int *
p00	0x0056f9dc {0}	int *
p01	0xcccccccc {???}	int *
p07	0xcccccccc {???}	int *
*p0	0	int
*p01	<Чтение памяти невозможно>	

Связь массивов и указателей (6)

```
32  #include <stdio.h>
33
34  void main() {
35      int a[8] = {0, 1, 2, 3, 4, 5, 6, 7}; // массив a содержащий 8 элементов целого типа
36
37      int* p0 = &a[0];
38      int* p1 = &a[1];
39      int* p7 = &a[7];
40
41      int* p00 = a;
42      int* p01 = a + 1;
43      int* p07 = a + 7;  ≤ 1 мс прошло
44
45      *p0 = 100;
46      *p01 = 200;
47  }
48
```

Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

Имя	Значение	Тип
a	0x0056f9dc {0, 1, 2, 3, 4, 5, 6, 7}	int[8]
[0]	0	int
[1]	1	int
[2]	2	int
[3]	3	int
[4]	4	int
[5]	5	int
[6]	6	int
[7]	7	int
a+1	0x0056f9e0 {1}	int *
a+7	0x0056f9f8 {7}	int *
p0	0x0056f9dc {0}	int *
p1	0x0056f9e0 {1}	int *
p7	0x0056f9f8 {7}	int *
p00	0x0056f9dc {0}	int *
p01	0x0056f9e0 {1}	int *
p07	0xcccccccc {???}	int *
*p0	0	int
*p01	1	int

Связь массивов и указателей (7)

```
32  #include <stdio.h>
33
34  void main() {
35      int a[8] = {0, 1, 2, 3, 4, 5, 6, 7}; // массив a содержащий 8 элементов целого типа
36
37      int* p0 = &a[0];
38      int* p1 = &a[1];
39      int* p7 = &a[7];
40
41      int* p00 = a;
42      int* p01 = a + 1;
43      int* p07 = a + 7;
44
45      *p0 = 100;  ≤ 1 мс прошло
46      *p01 = 200;
47  }
48
```

Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

Имя	Значение	Тип
a	0x0056f9dc {0, 1, 2, 3, 4, 5, 6, 7}	int[8]
[0]	0	int
[1]	1	int
[2]	2	int
[3]	3	int
[4]	4	int
[5]	5	int
[6]	6	int
[7]	7	int
a+1	0x0056f9e0 {1}	int *
a+7	0x0056f9f8 {7}	int *
p0	0x0056f9dc {0}	int *
p1	0x0056f9e0 {1}	int *
p7	0x0056f9f8 {7}	int *
p00	0x0056f9dc {0}	int *
p01	0x0056f9e0 {1}	int *
p07	0x0056f9f8 {7}	int *
*p0	0	int
*p01	1	int

Связь массивов и указателей (8)

```
32  #include <stdio.h>
33
34  void main() {
35      int a[8] = {0, 1, 2, 3, 4, 5, 6, 7}; // массив a содержащий 8 элементов целого типа
36
37      int* p0 = &a[0];
38      int* p1 = &a[1];
39      int* p7 = &a[7];
40
41      int* p00 = a;
42      int* p01 = a + 1;
43      int* p07 = a + 7;
44
45      *p0 = 100;
46      *p01 = 200;  ≤ 1 мс прошло
47  }
48
```

Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

Имя	Значение	Тип
a	0x0056f9dc {100, 1, 2, 3, 4, 5, 6, 7}	int[8]
[0]	100	int
[1]	1	int
[2]	2	int
[3]	3	int
[4]	4	int
[5]	5	int
[6]	6	int
[7]	7	int
a+1	0x0056f9e0 {1}	int *
a+7	0x0056f9f8 {7}	int *
p0	0x0056f9dc {100}	int *
p1	0x0056f9e0 {1}	int *
p7	0x0056f9f8 {7}	int *
p00	0x0056f9dc {100}	int *
p01	0x0056f9e0 {1}	int *
p07	0x0056f9f8 {7}	int *
*p0	100	int
*p01	1	int

Связь массивов и указателей (9)

```
32  #include <stdio.h>
33
34  void main() {
35      int a[8] = {0, 1, 2, 3, 4, 5, 6, 7}; // массив a содержащий 8 элементов целого типа
36
37      int* p0 = &a[0];
38      int* p1 = &a[1];
39      int* p7 = &a[7];
40
41      int* p00 = a;
42      int* p01 = a + 1;
43      int* p07 = a + 7;
44
45      *p0 = 100;
46      *p01 = 200;
47  } 1 мс прошло
48
```

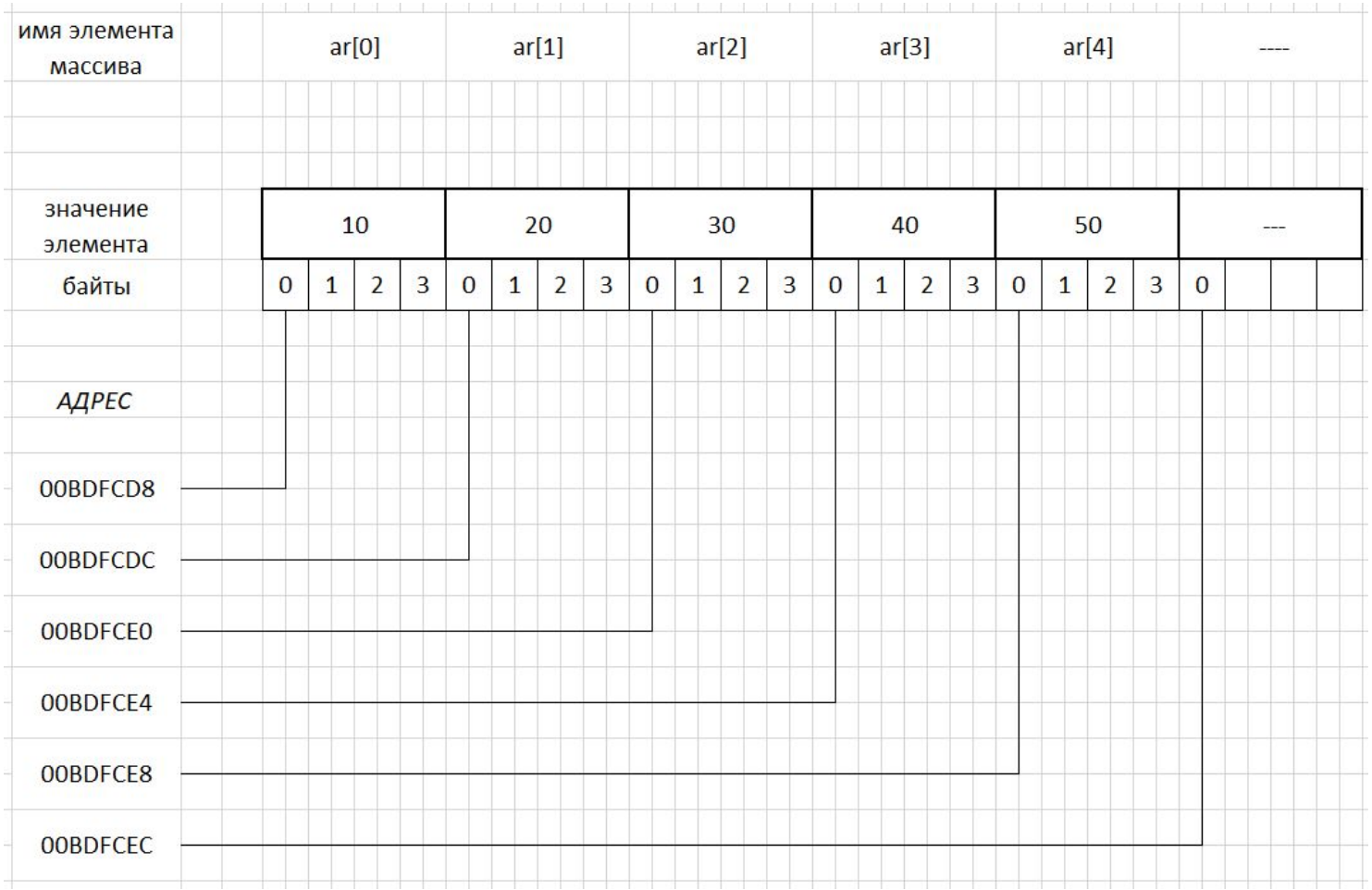
Контрольные значения 1

Поиск (Ctrl+E) Глубина поиска: 3

Имя	Значение	Тип
a	0x0056f9dc {100, 200, 2, 3, 4, 5, 6, 7}	int[8]
[0]	100	int
[1]	200	int
[2]	2	int
[3]	3	int
[4]	4	int
[5]	5	int
[6]	6	int
[7]	7	int
a+1	0x0056f9e0 {200}	int *
a+7	0x0056f9f8 {7}	int *
p0	0x0056f9dc {100}	int *
p1	0x0056f9e0 {200}	int *
p7	0x0056f9f8 {7}	int *
p00	0x0056f9dc {100}	int *
p01	0x0056f9e0 {200}	int *
p07	0x0056f9f8 {7}	int *
*p0	100	int
*p01	200	int

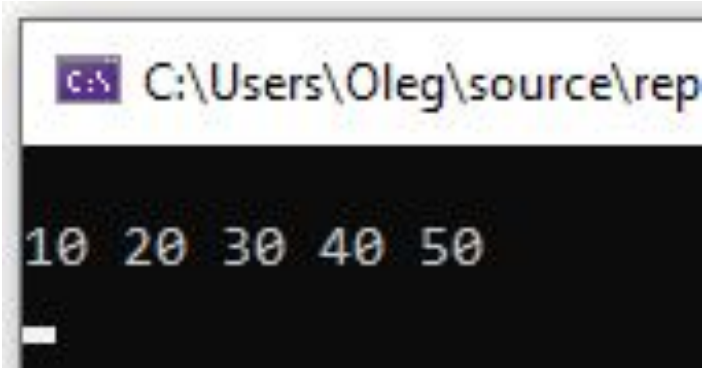
Массив ar1

```
#include <stdio.h>
void main() {
    int ar[5] = { 10, 20, 30, 40, 50 };
```



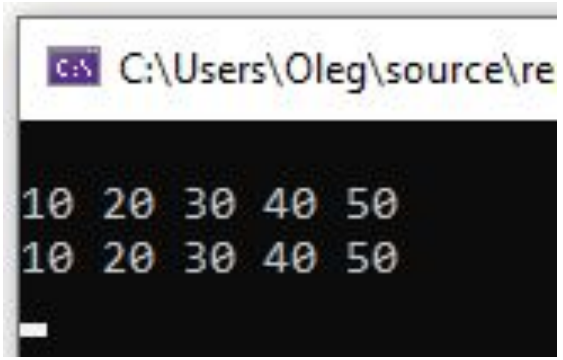
Вывод массива

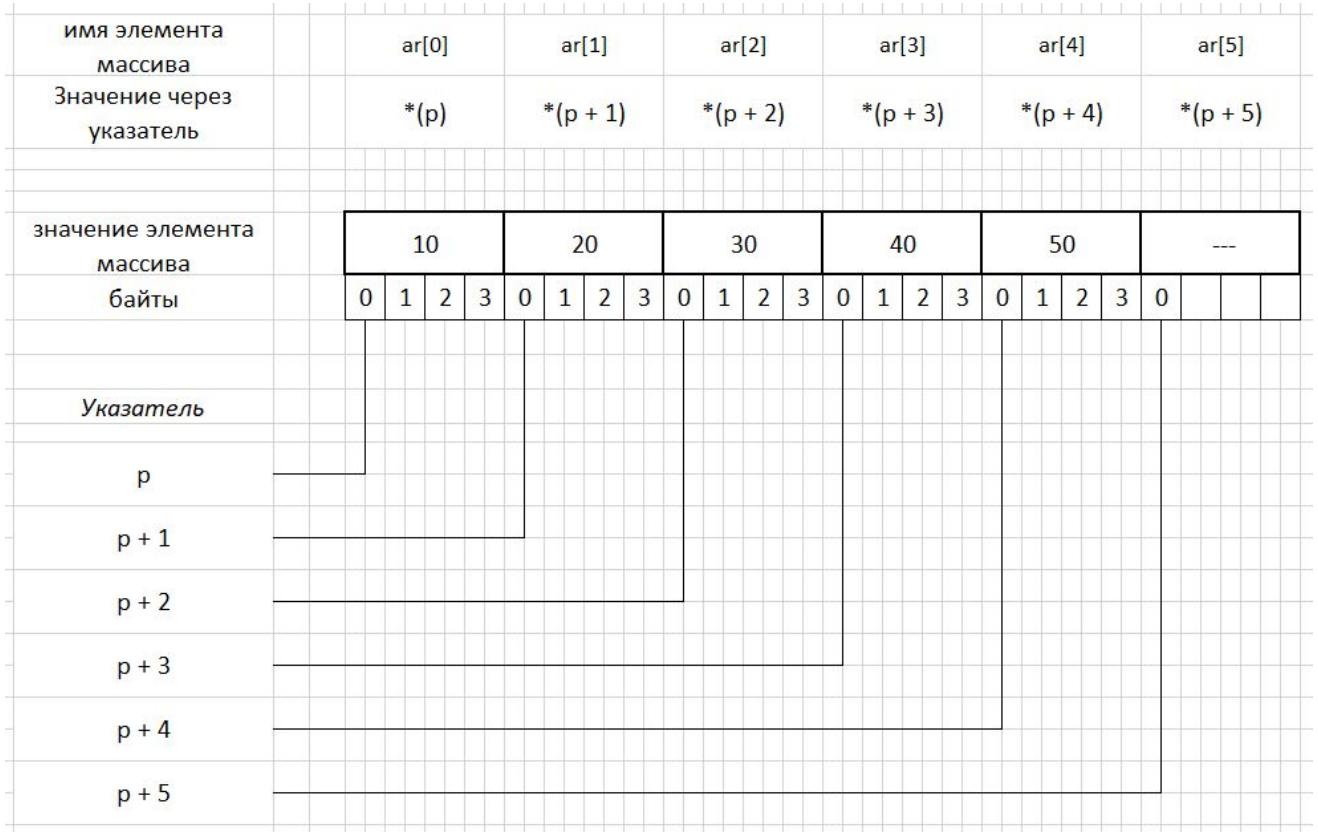
```
int i;

printf("\n");

for (i = 0; i < 5; i++)
    printf("%d ", ar[i]);
printf("\n");
```



Вывод массива при помощи указателей

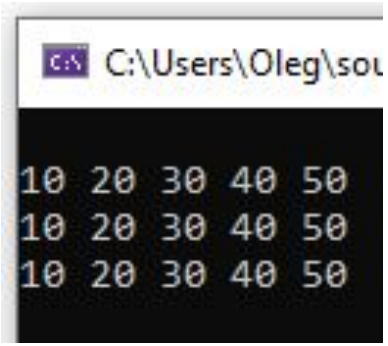
```
int* p;
p = ar;
for (i = 0; i < 5; i++)
        printf("%d ", *(p + i));
printf("\n");
```

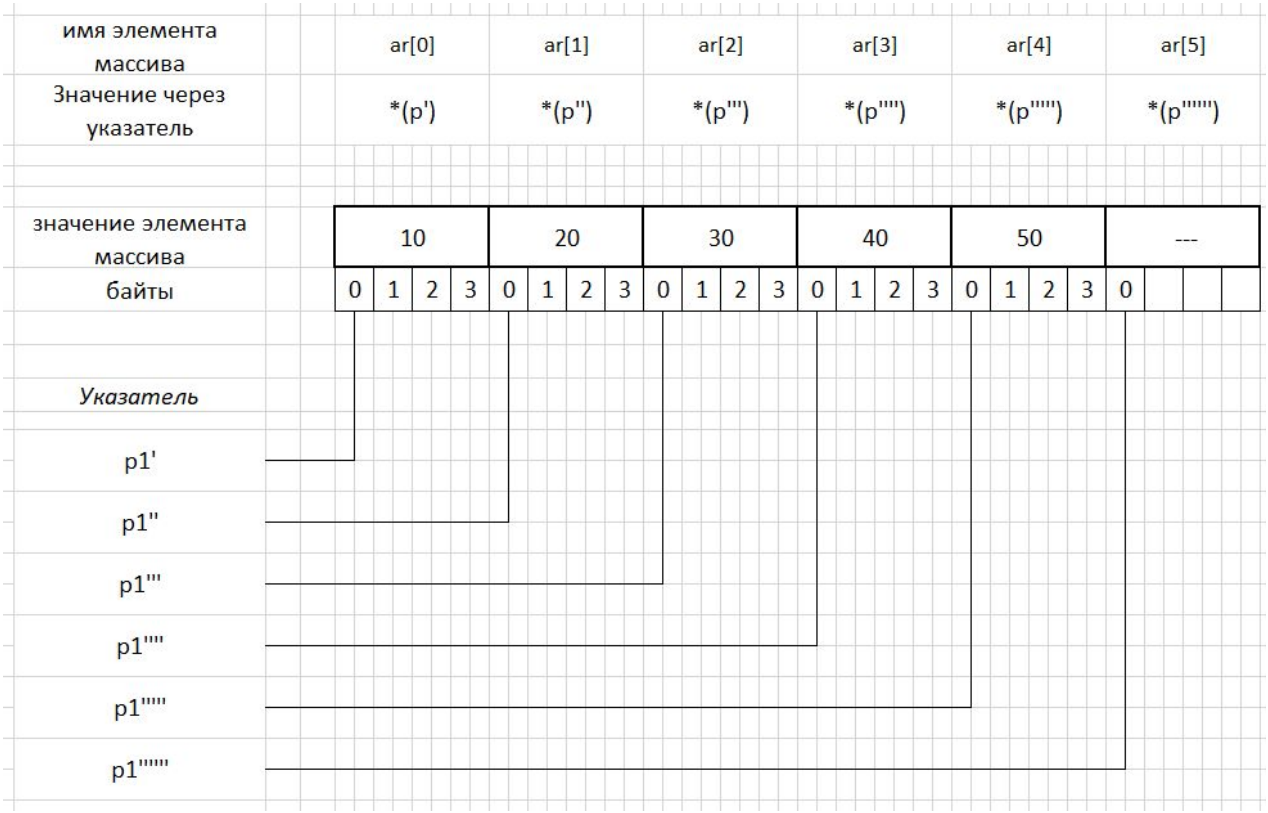




Вывод массива при помощи указателей (v 2)

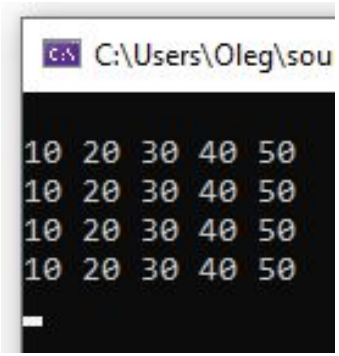
```
int* p1;

p1 = ar;
for (i = 0; i < 5; i++)
    printf("%d ", *(p1++));
printf("\n");
```

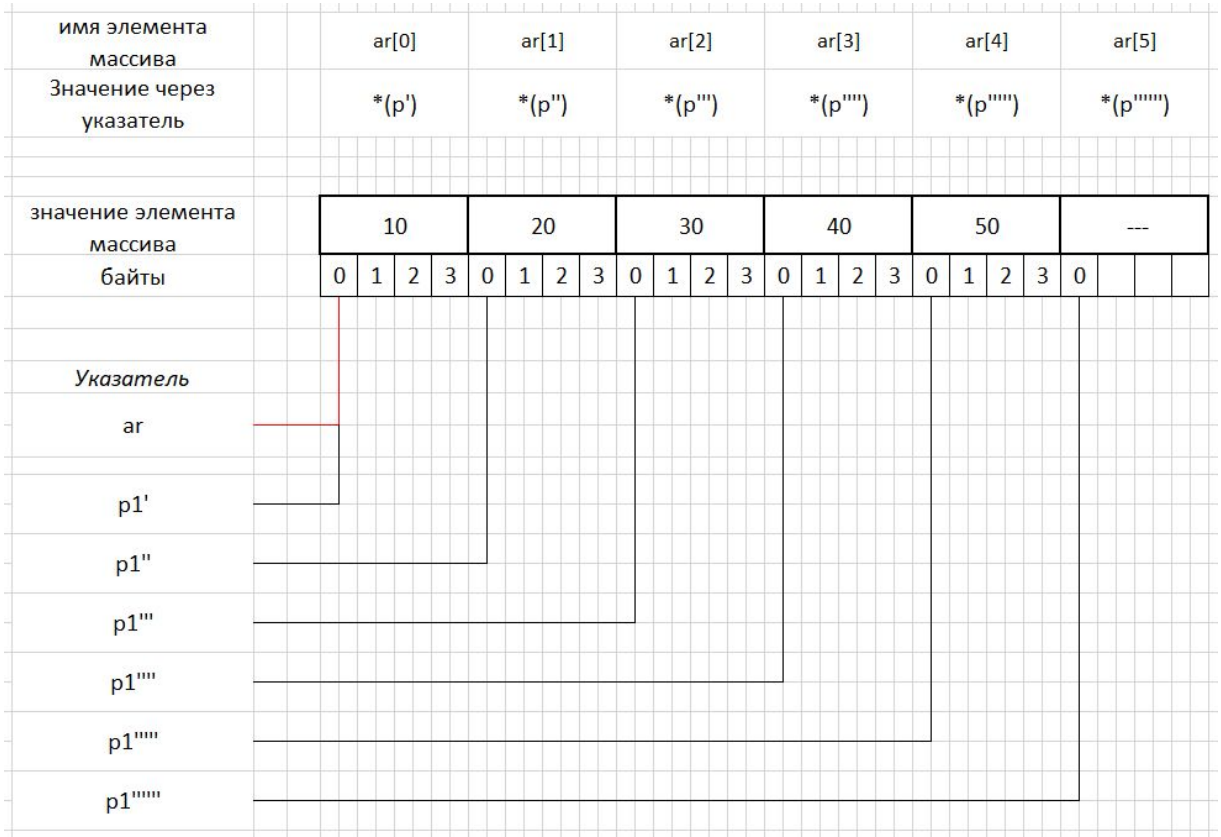




Вывод массива при помощи указателей (v 3)

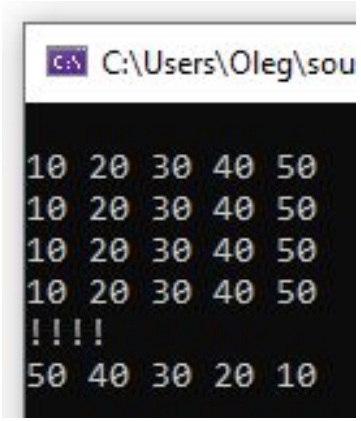
```
p1 = ar;

while (p1 - ar < 5)
    printf("%d ", *(p1++));
printf("\n");
```

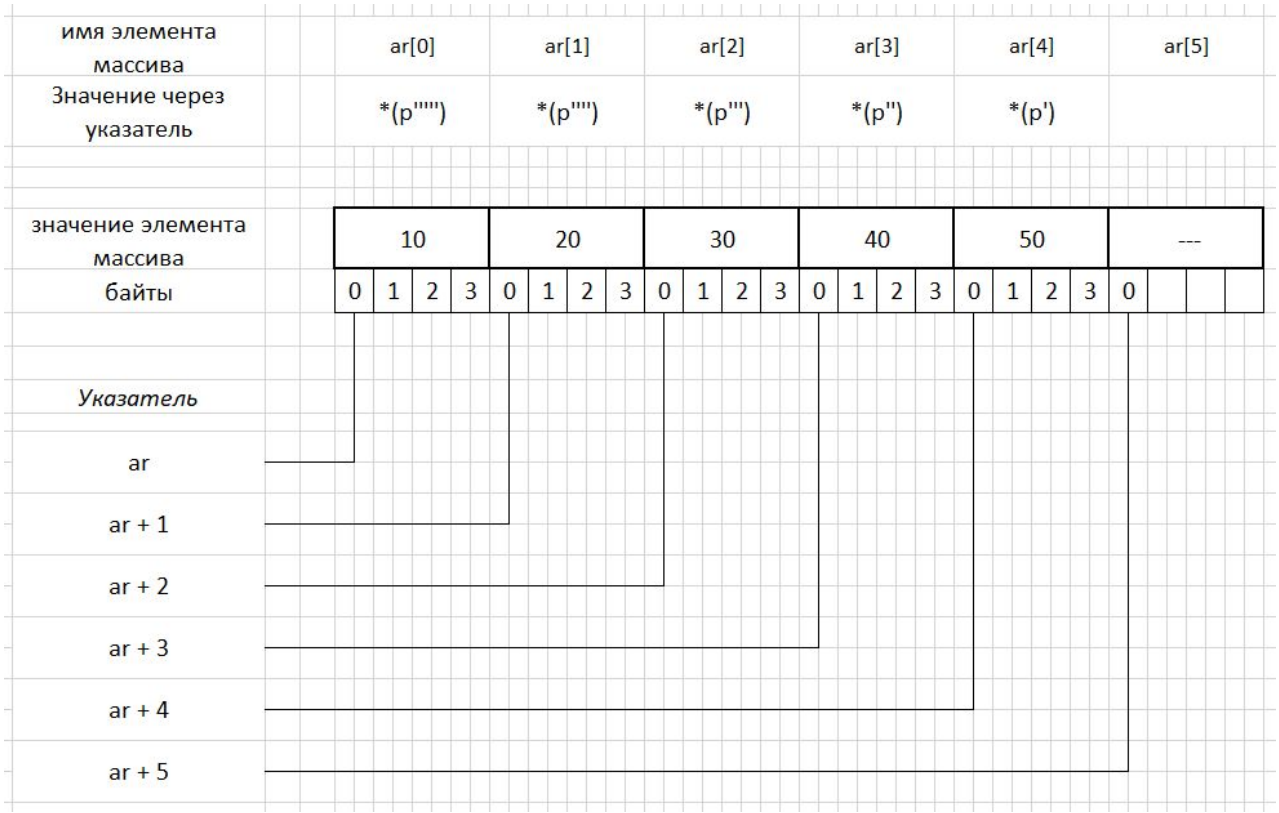




Нечто особое

```
printf("!!!!\n");
p1 = ar + 4;
while (p1 >= ar)
    printf("%d ", *(p1--));
printf("\n");
```



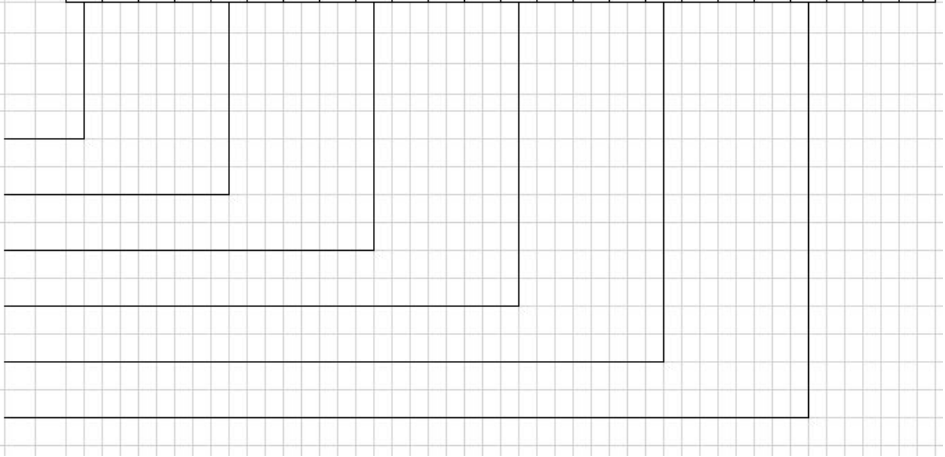


Нечто совсем особое

```
printf("\n");

for (i = 0; i < 5; i++)
    printf("%d ", i[ar]);
printf("\n");
```

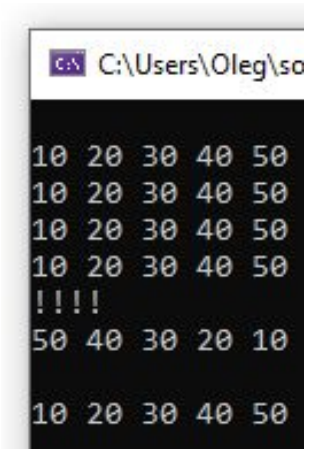
```
C:\Users\Oleg\so

10 20 30 40 50
10 20 30 40 50
10 20 30 40 50
10 20 30 40 50
!!!!
50 40 30 20 10

10 20 30 40 50
```

адрес элемента = указатель на элемент	ar	ar + 1	ar + 2	ar + 3	ar + 4	ar + 5
значение через указатель	*(ar + 0)	*(ar + 1)	*(ar + 2)	*(ar + 3)	*(ar + 4)	*(ar + 5)
значение через указатель v2	*(0 + ar)	*(1 + ar)	*(2 + ar)	*(3 + ar)	*(4 + ar)	*(5 + ar)
значение через индекс	*(ar + 0) = ar[0]	*(ar + 1) = ar[1]	*(ar + 2) = ar[2]	*(ar + 3) = ar[3]	*(ar + 4) = ar[4]	*(ar + 5) = ar[5]
значение через индекс v2	***(0 + ar) = 0[ar]**	***(1 + ar) = 1[ar]**	***(2 + ar) = 2[ar]**	***(3 + ar) = 3[ar]**	***(4 + ar) = 4[ar]**	***(5 + ar) = 5[ar]**
значение элемента массива	10	20	30	40	50	---
байты	0 1 2 3	0 1 2 3	0 1 2 3	0 1 2 3	0 1 2 3	0



Нечто совсем особое

```
printf("\n");

for (i = 0; i < 5; i++)
    printf("%d ", i[ar]);
printf("\n");
```



адрес элемента = указатель на элемент	ar	ar + 1	ar + 2	ar + 3	ar + 4	ar + 5
значение через указатель v2	*(0 + ar)	*(1 + ar)	*(2 + ar)	*(3 + ar)	*(4 + ar)	*(5 + ar)
значение через индекс v2	**0[ar]**	**1[ar]**	**2[ar]**	**3[ar]**	**4[ar]**	**5[ar]**
значение элемента массива	10	20	30	40	50	---
байты	0 1 2 3	0 1 2 3	0 1 2 3	0 1 2 3	0 1 2 3	0

Массивы и указатели – еще раз

```
int a[8] = {0, 1, 2, 3, 4, 5, 6, 7};
```

a ≡ &a[0]

a + 1 ≡ &a[1] □ *(a + 1) ≡ a[1]

a + 1 ≡ 1 + a □ a[1] ≡ *(a + 1) ≡ *(1 + a) ≡ 1[a]

a + i ≡ &a[i] □ *(a + i) ≡ a[i]

a + i ≡ i + a □ a[i] ≡ *(a + i) ≡ *(i + a) ≡ i[a]

Операции над указателями

```
#include <stdio.h>

void main() {
    int a[8] = {10, 20, 30, 40, 50, 60, 70, 80};

    int* p1 = &a[1];
    int* p2 = &a[7];

    int* p3 = p1 + 2;   //1: К указателю добавляем целое число
    int* p4 = p2 - 3;   //2: Из указателя вычитаем целое число
    int  d = p2 - p1;   //3: Из одного указателя вычитаем другой

    printf("%d %d %d %d %d", *p1, *p2, *p3, *p4, d);
}
```

Что можно почитать про «Указатели и массивы»

Курс Лекций по Языку Си. Указатели и массивы -
https://learnc.info/c/arrays_vs_pointers.html

Язык программирования Си. Глава 5. Указатели и массивы -
https://cpp.com.ru/kr_cbook/ch5kr.html

Использование указателей при обработке строк

Собственная реализация strlen v2

```
int strlen_my(char s[])
{
    char* p = s;
    while (*p++);
    return p - s - 1;
}
```

```
void main() {
    char s[10] = "Hi!";
    printf("len = %d\n", strlen_my(s));

    s[3] = ' '; s[4] = '\0';
    printf("len = %d\n", strlen_my(s));

    s[4] = 'W'; s[5] = 'o'; s[6] = 'r'; s[7] = 'l';
    s[8] = 'd'; s[9] = '\0';
    printf("len = %d\n", strlen_my(s));
}
```

Собственная реализация strlen v2 (2)

```
int strlen_my(char s[])
{
    char* p = s;
    while (*p++);
    return p - s - 1;
}
```

```
char s[10] = "Hi!";
printf("len = %d\n", strlen_my(s));
```

Имя элемента	s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]	s[10]
Адрес элемента	s	s + 1	s + 2	s + 3	s + 4	s + 5	s + 6	s + 7	s + 8	s + 9	s + 10
Значение элемента	H	i	!	\0							

Собственная реализация strlen v2 (3)

```
int strlen_my(char s[])
{
    char* p = s;
    while (*p++);
    return p - s - 1;
}
```

```
char s[10] = "Hi!";
s[3] = ' '; s[4] = '\0';
printf("len = %d\n", strlen_my(s));
```

Имя элемента	s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]	s[10]
Адрес элемента	s	s + 1	s + 2	s + 3	s + 4	s + 5	s + 6	s + 7	s + 8	s + 9	s + 10
Значение элемента	H	i	!		\0						

Собственная реализация strlen v2 (4)

```
int strlen_my(char s[])
{
    char* p = s;
    while (*p++);
    return p - s - 1;
}
```

```
char s[10] = "Hi!";
s[3] = ' '; s[4] = '\0';
s[4] = 'W'; s[5] = 'o'; s[6] = 'r'; s[7] = 'l';
s[8] = 'd'; s[9] = '\0';
printf("len = %d\n", strlen_my(s));
```

Имя элемента	s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]	s[10]
Адрес элемента	s	s + 1	s + 2	s + 3	s + 4	s + 5	s + 6	s + 7	s + 8	s + 9	s + 10
Значение элемента	H	i	!		W	o	r	l	d	\0	

Собственная реализация strcpy

```
int strcpy_my(char* dest, char* src)
{
   while (*dest++ = *src++);
}
```

```
void main() {
      char src[] = "Button";
      char dest[10];

      printf("src = %s, dest = %s\n", src, dest);

      strcpy_my(dest, src);
      printf("src = %s, dest = %s\n", src, dest)n;
}
```

Собственная реализация strcpy (2)

```
int strcpy_my(char* dest, char* src)
{
   while (*dest++ = *src++);
}
```

char src[] = "Button";
char dest[10];

strcpy_my(dest, src);

Имя элемента	src[0]	src[1]	src[2]	src[3]	src[4]	src[5]	src[6]	src[7]
Адрес элемента	src	src + 1	src + 2	src + 3	src + 4	src + 5	src + 6	src + 7
Значение элемента	B	u	t	t	o	n	\0	

Имя элемента	dest[0]	dest[1]	dest[2]	dest[3]	dest[4]	dest[5]	dest[6]	dest[7]	dest[8]	dest[9]	dest[10]
Адрес элемента	dest	dest + 1	dest + 2	dest + 3	dest + 4	dest + 5	dest + 6	dest + 7	dest + 8	dest + 9	dest + 10
Значение элемента	???	???	???	???	???	???	???	???	???	???	???

Собственная реализация strcat

```
int strcat_my(char* dest, char* src)
{
   while (*dest) dest++;
   while (*dest++ = *src++);
}
```

```
void main() {
      char src[] = "Button";
      char dest[10] = "<>";
      printf("src = %s, dest = %s\n", src, dest);

      strcat_my(dest, src);
      printf("src = %s, dest = %s\n", src, dest);

      strcat_my(dest, "!");
      printf("src = %s, dest = %s\n", src, dest);
}
```

Собственная реализация strcat (2)

```
int strcat_my(char* dest, char* src)
{
   while (*dest) dest++;
   while (*dest++ = *src++);
}
```

char src[] = "Button";
char dest[10] = "<>";

strcat_my(dest, src);
strcat_my(dest, "!");

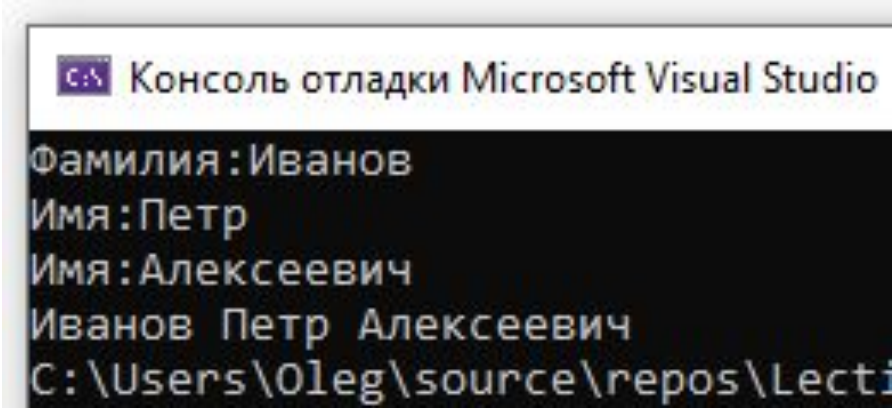
Имя элемента	src[0]	src[1]	src[2]	src[3]	src[4]	src[5]	src[6]	src[7]
Адрес элемента	src	src + 1	src + 2	src + 3	src + 4	src + 5	src + 6	src + 7
Значение элемента	B	u	t	t	o	n	\0	

Имя элемента	dest[0]	dest[1]	dest[2]	dest[3]	dest[4]	dest[5]	dest[6]	dest[7]	dest[8]	dest[9]	dest[10]
Адрес элемента	dest	dest + 1	dest + 2	dest + 3	dest + 4	dest + 5	dest + 6	dest + 7	dest + 8	dest + 9	dest + 10
Значение элемента	<	>	\0	???	???	???	???	???	???	???	???

Ввод строк с клавиатуры

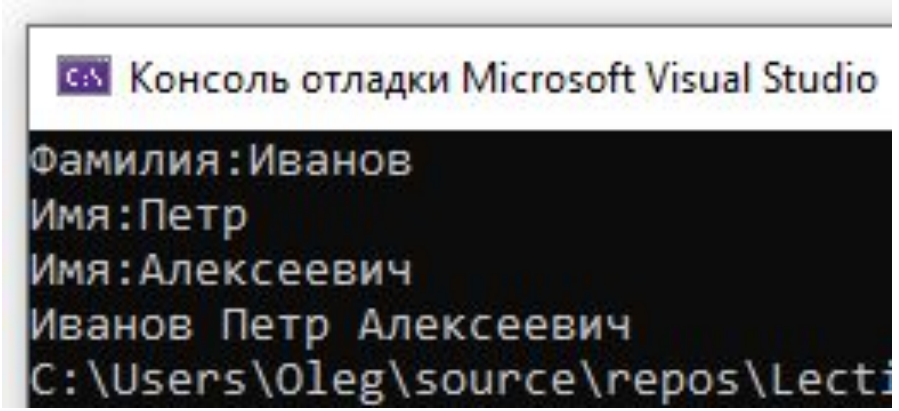
Ввод при помощи scanf

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <Windows.h>

void main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    char f[30], i[30], o[30];

    printf("Фамилия:");
    scanf("%s", f);

    printf("Имя:");
    scanf("%s", i);

    printf("Имя:");
    scanf("%s", o);

    printf("%s %s %s", f, i, o);
}
```



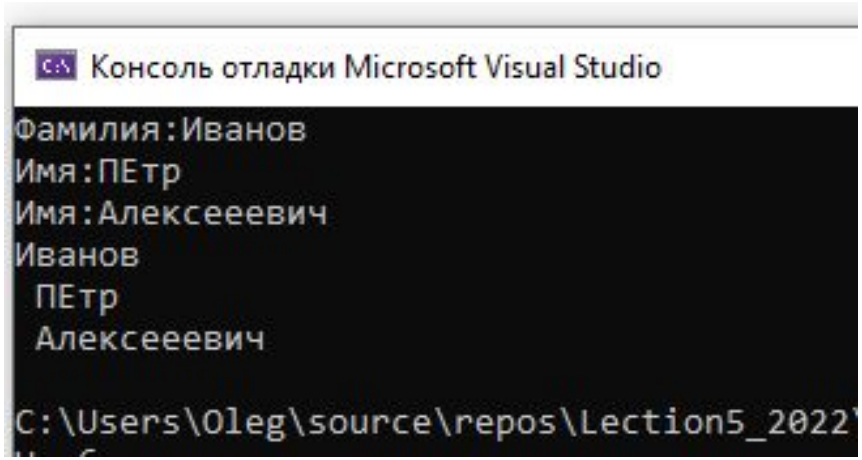
Ввод при помощи scanf_s

```
//#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <Windows.h>

void main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    char f[30], i[30], o[30];

    printf("Фамилия:");
    scanf_s("%s", f, 30);

    printf("Имя:");
    scanf_s("%s", i, 30);

    printf("Имя:");
    scanf_s("%s", o, 30);

    printf("%s %s %s", f, i, o);
}
```



Ввод при помощи fgets

```c
#include <stdio.h>
#include <Windows.h>

void  main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    char f[30], i[30], o[30];

    printf("Фамилия:");
    fgets(f, 30, stdin);

    printf("Имя:");
    fgets(i, 30, stdin);

    printf("Имя:");
    fgets(o, 30, stdin);

    printf("%s %s %s", f, i, o);
}
```

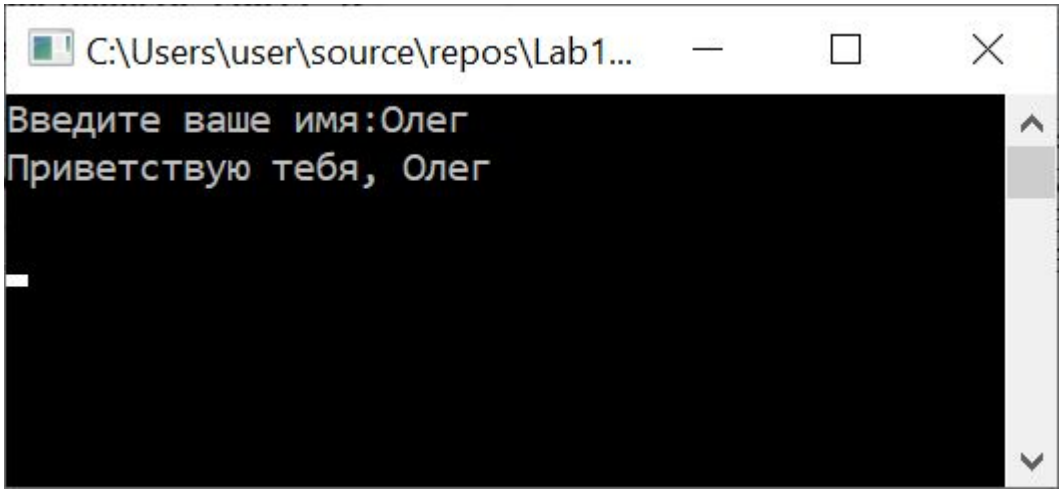


Лабораторная работа №20

Знакомство с обработкой строк и символов

Задача 1 – Hello по русски!

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <Windows.h>

void main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    char name[12];
    printf("Введите ваше имя:");
    fgets(name, 11, stdin);
    printf("Приветствую тебя, %s\n", name);

    {
        int x;
        scanf("%d", &x);
    }
}
```

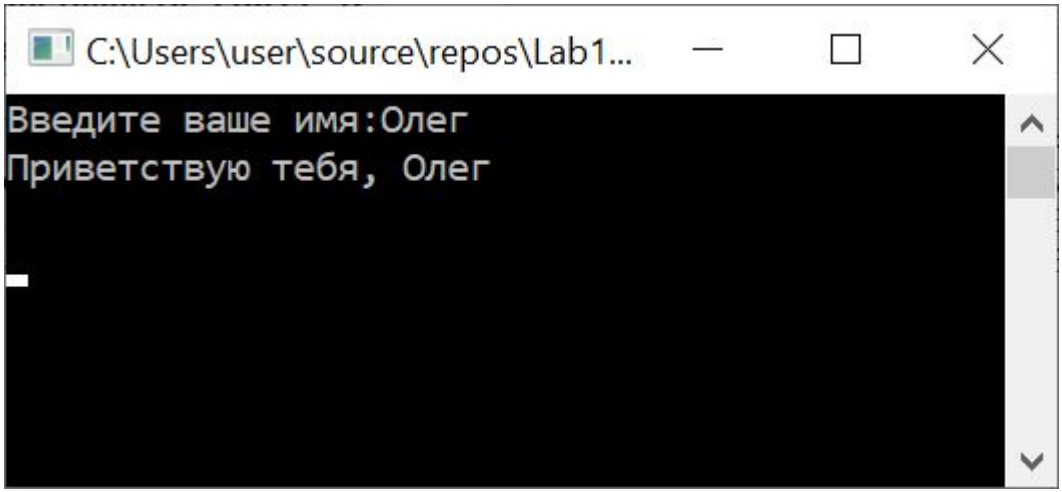


Задача 1+ – Hello по русски!

1. Закомментируйте строку SetConsoleCP(1251);
   Запустите программу – посмотрите что получилось.
   Раскомментируйте!

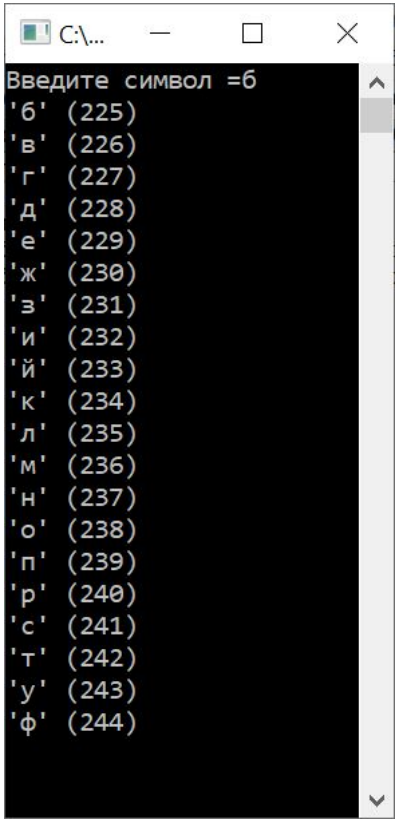
2. Закомментируйте строку SetConsoleOutputCP (1251);
   Запустите программу – посмотрите что получилось.
   Раскомментируйте!

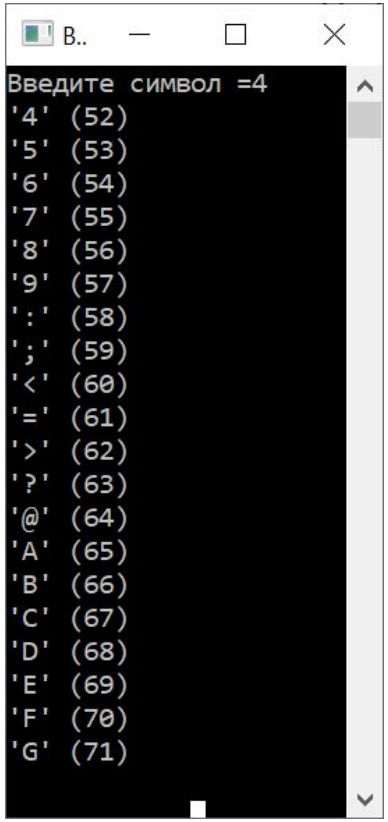
3. Попробуйте ввести длинное имя – «Иван Иванович Иванов»
   Посмотрите что получилось? Почему так получилось?



Задача 2

Ввести символ CH
Вывести на экран 20 символов с кодами от CH до CH+19



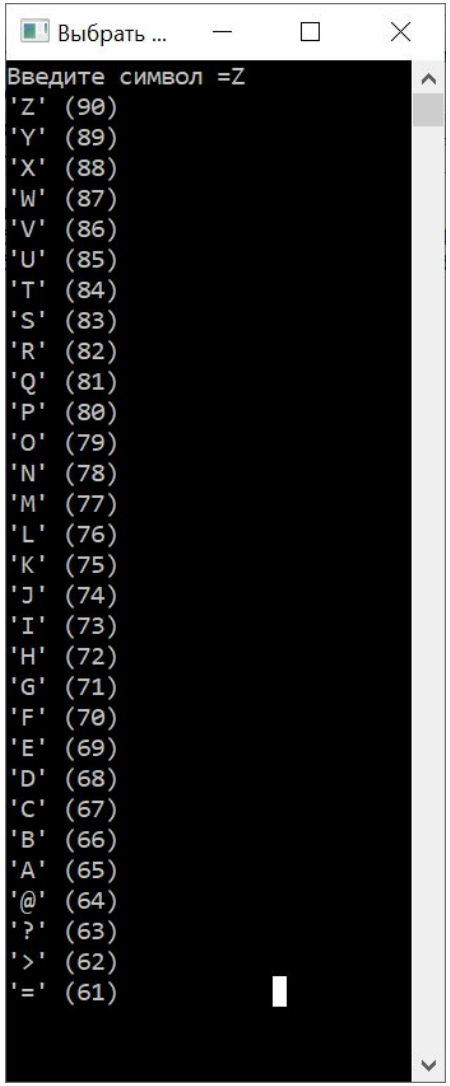


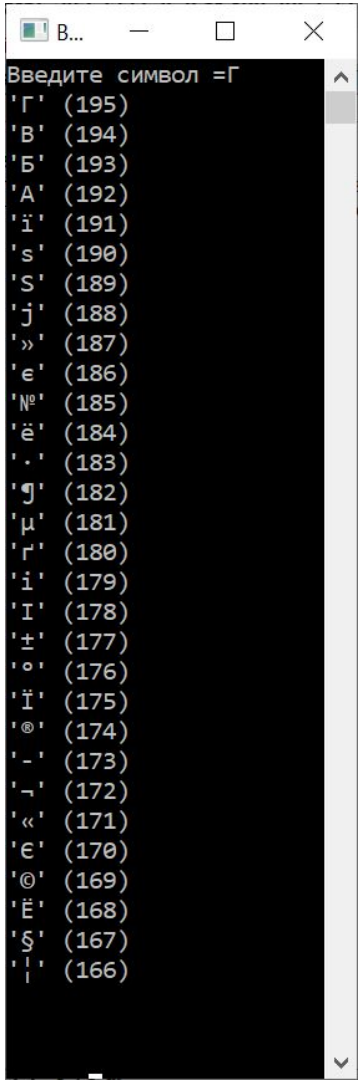

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <Windows.h>

void main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    printf("Введите символ =");
    int ch = getchar();

    for (int c = ch; c <= ch + 19; c++) {
        printf("'%c' (%d)\n", c, c);
    }
    printf("\n\n\n");

    {
        int x;
        scanf("%d", &x);
    }
}
```

Задача 3*

Ввести символ CH

Вывести на экран 30 символов с кодами от CH до CH-29

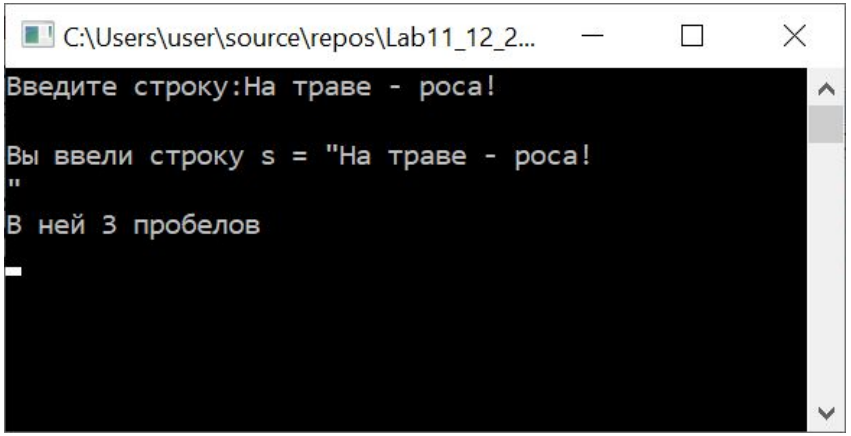




Задача 4

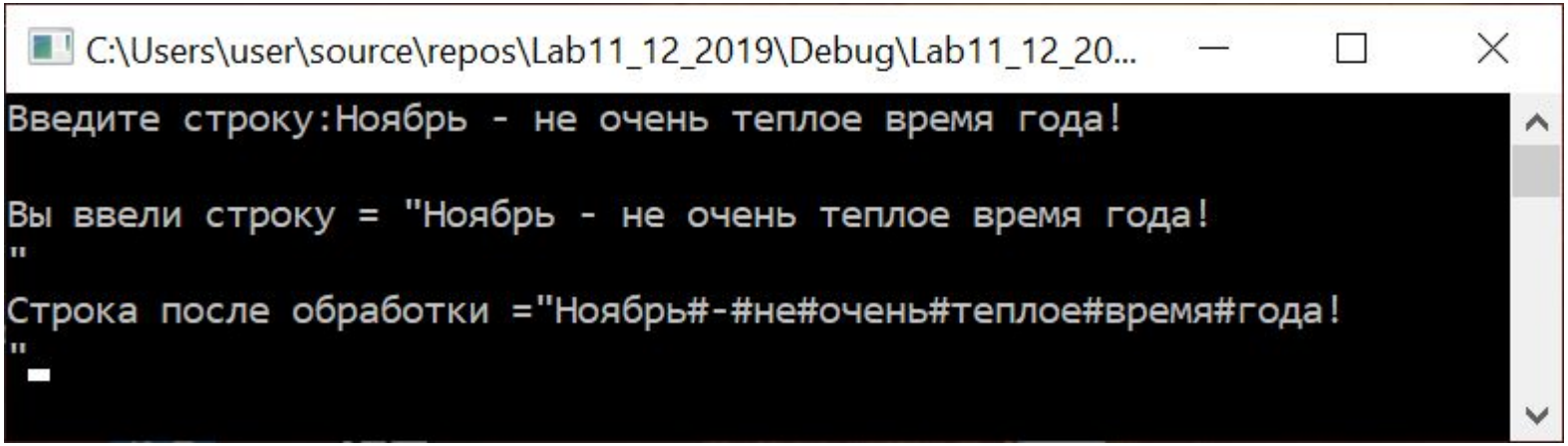
Ввести строку s. Подсчитать, сколько в ней пробелов

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <Windows.h>

void main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    // ввод строки
    char s[80];
    printf("Введите строку:");
    fgets(s, 79, stdin);

    printf("\nВы ввели строку s = \"%s\"", s);

    int cnt = 0;
    for (int i = 0; i < strlen(s); i++) {
        if (s[i] == ' ') cnt++;
    }

    printf("\nВ ней %d пробелов\n", cnt);

    {
        int x;
        scanf("%d", &x);
    }
}
```



Задача 5*

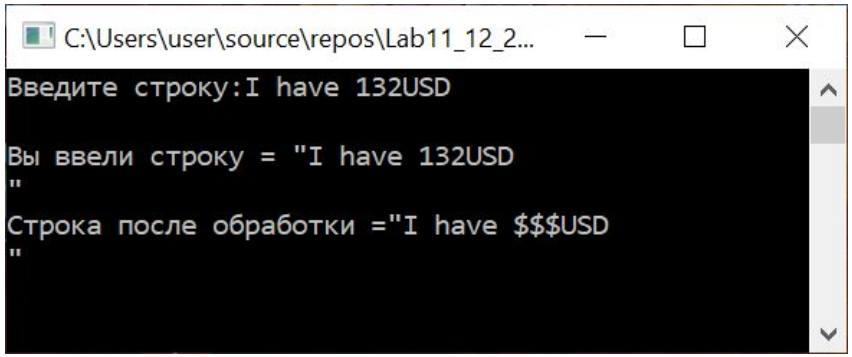
Ввести строку s. Все пробелы в ней заменить символом ‘#’.
После обработки получившуюся строку вывести в консоль

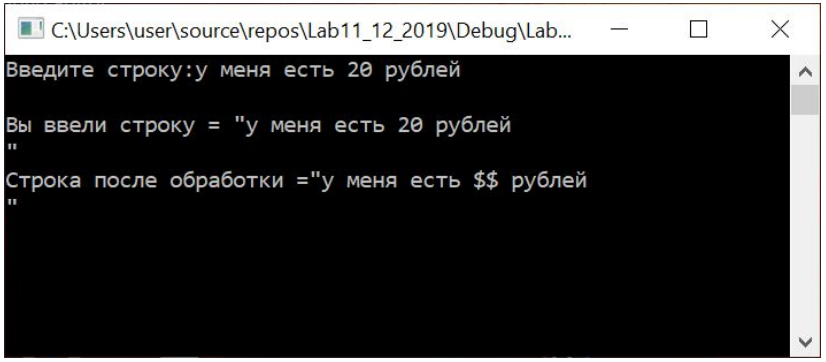


Задача 6.1

Ввести строку s. Все цифры в ней заменить символом ‘$‘.
Используйте стандартную функцию isdigit()

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <ctype.h>
#include <Windows.h>


// Задача  6
// Ввести строку s. Все цифры в ней заменить символом '$'
for (int i = 0; s[i] != '\0'; i++) {
    if (isdigit(s[i]))
        s[i] = '$';
}
```



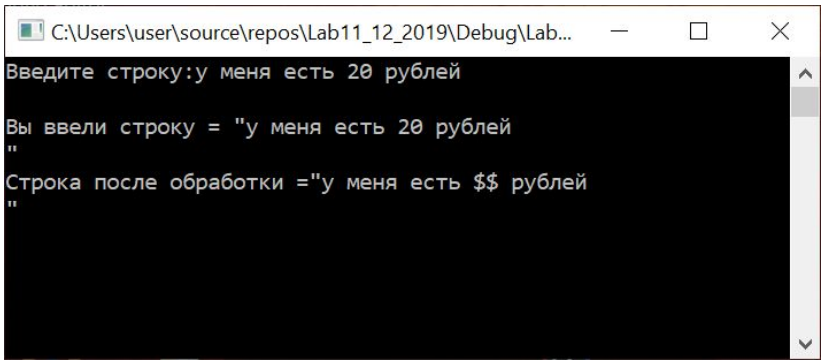


Задача 6.2

Ввести строку s. Все цифры в ней заменить символом ‘$‘
Используйте свою собственную реализацию функции isdigit()

```
int isDigitMy(char c) {
    if (c >= '0' && c <= '9')
        return 1;
    return 0;
}
```

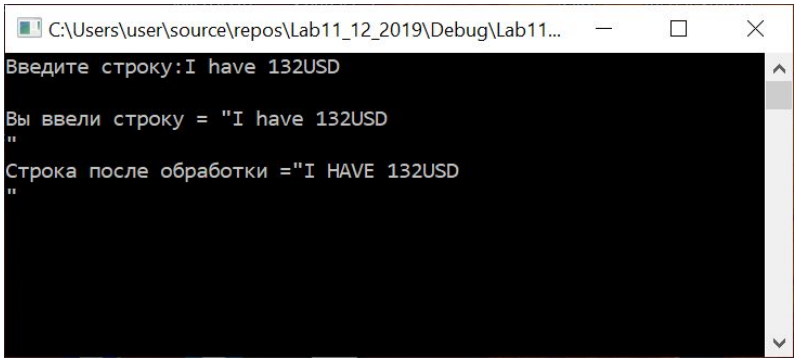
```
// Задача  6
// Ввести строку s. Все цифры в ней заменить символом '$'
for (int i = 0; s[i] != '\0'; i++) {
    //if (isdigit(s[i]))
    if (isDigitMy(s[i]))
        s[i] = '$';
}
```



Задача 7.1

Ввести строку s (без русских символов). Все маленькие латинские буквы превратить в большие
Используйте стандартную функцию toupper

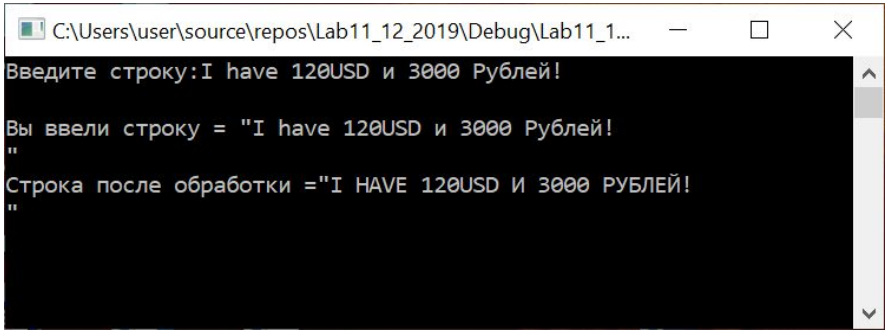
```
for (int i = 0; s[i] != '\0'; i++) {
    s[i] = toupper(s[i]);
}
```



Задача 7.2*

Ввести строку s с русскими и латинскими символами. Все маленькие буквы превратить в большие.
Используйте свою собственную реализацию функции toupper

```
int toUpperAll(int c) {
    int new_c = c;
    if (c >= 'a' && c <= 'z')
        new_c = 'A' + (c - 'a');
    if (c >= 'a' && c <= 'п')
        new_c = 'A' + (c - 'a');
    if (c >= 'р' && c <= 'я')
        new_c = 'Р' + (c - 'р');
    if (c == 'ё')
        new_c = 'Ё';
    return new_c;
}
```

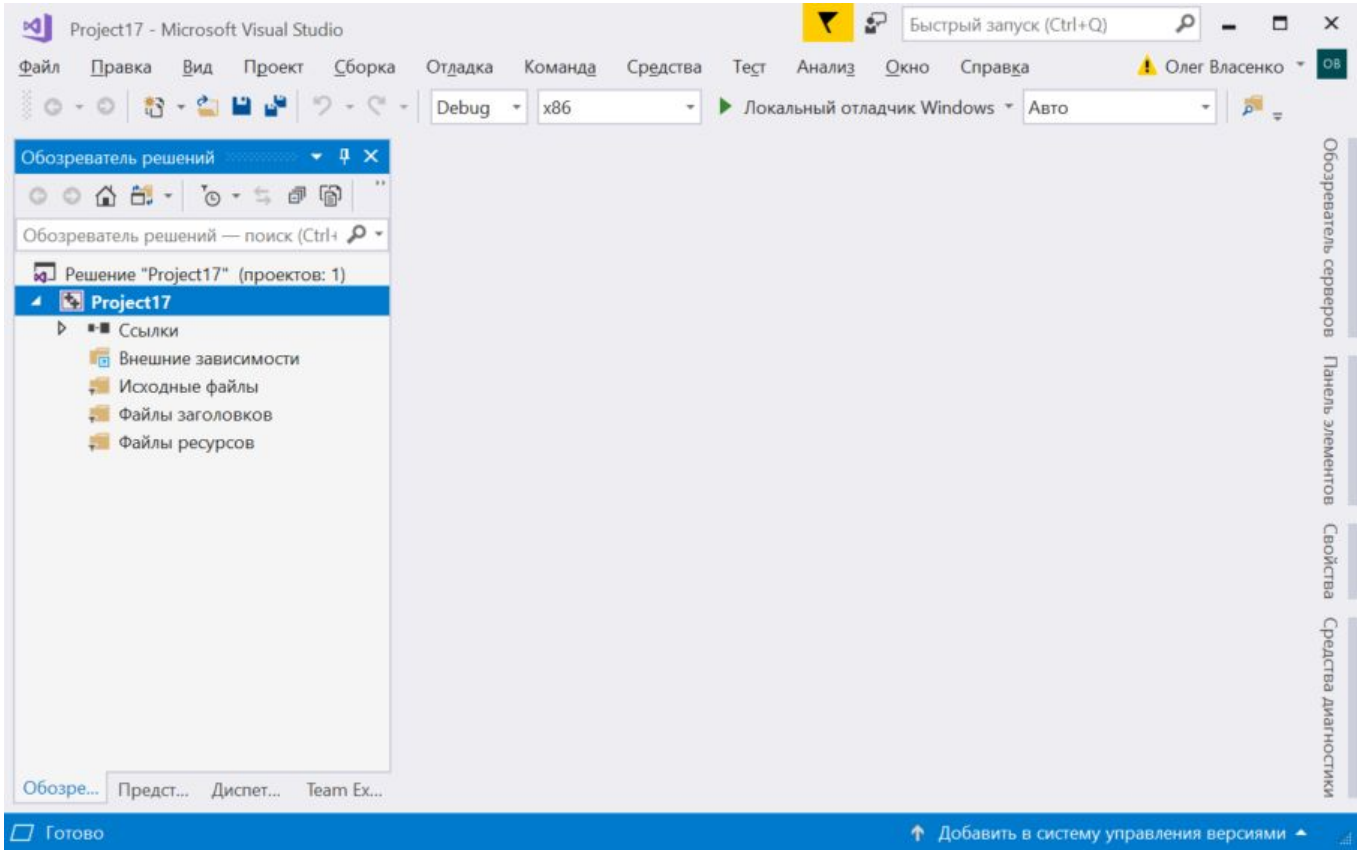


Задача 8*

Создать на основе разработанного кода проект с собственным модулем.

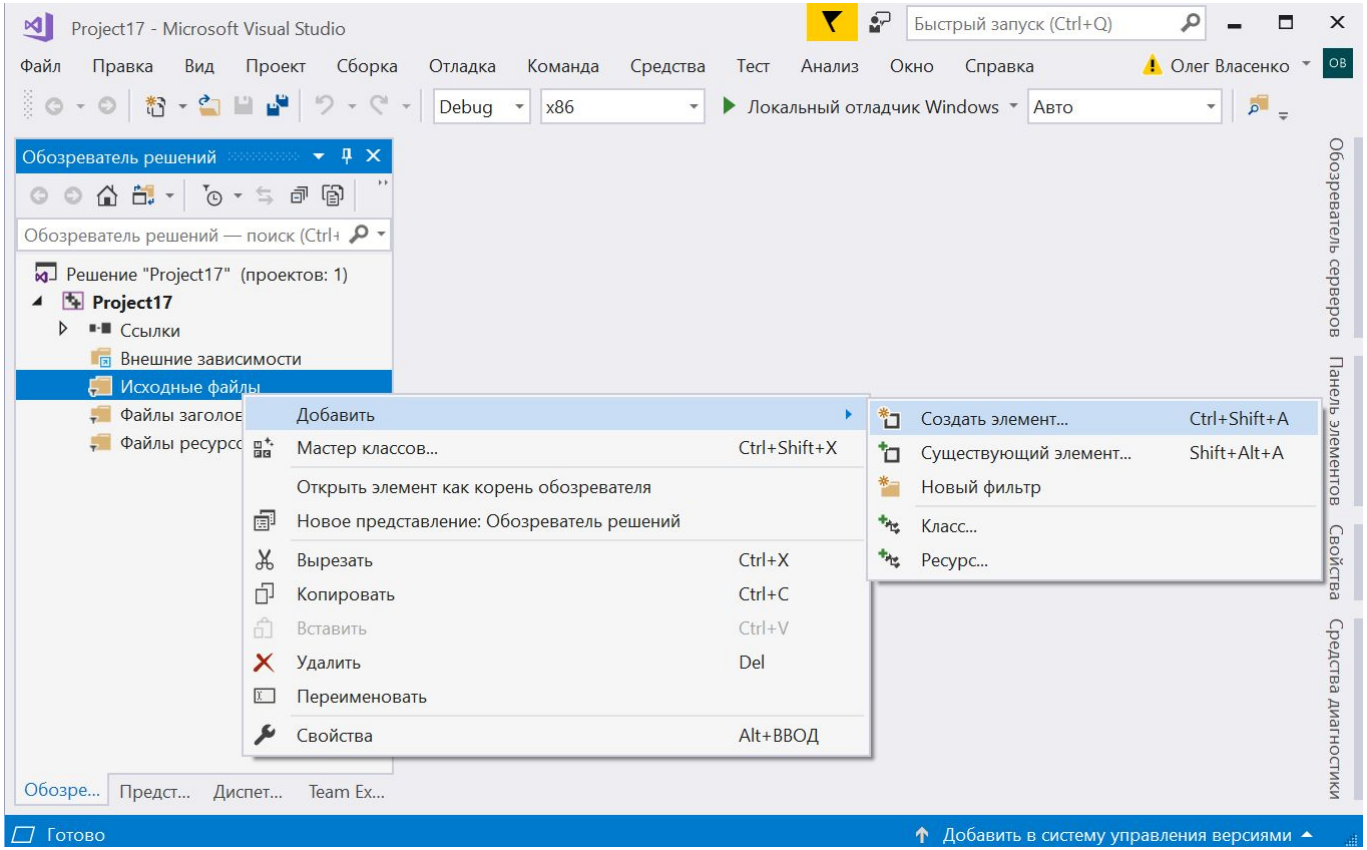
Задача 8* (1)

Создаем новый проект



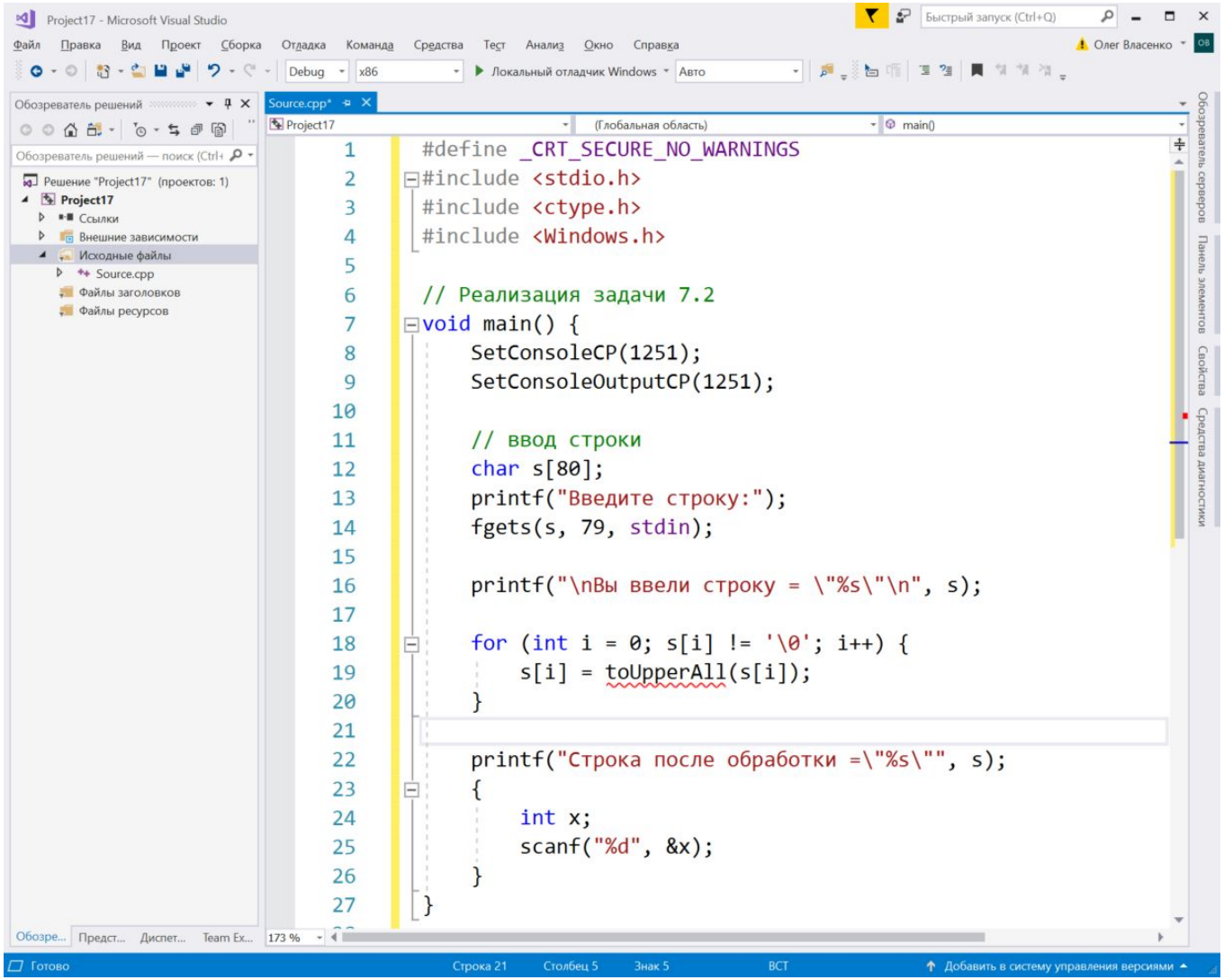
Задача 8* (2)

Создаем файл для главного модуля – в нем будет находиться main()



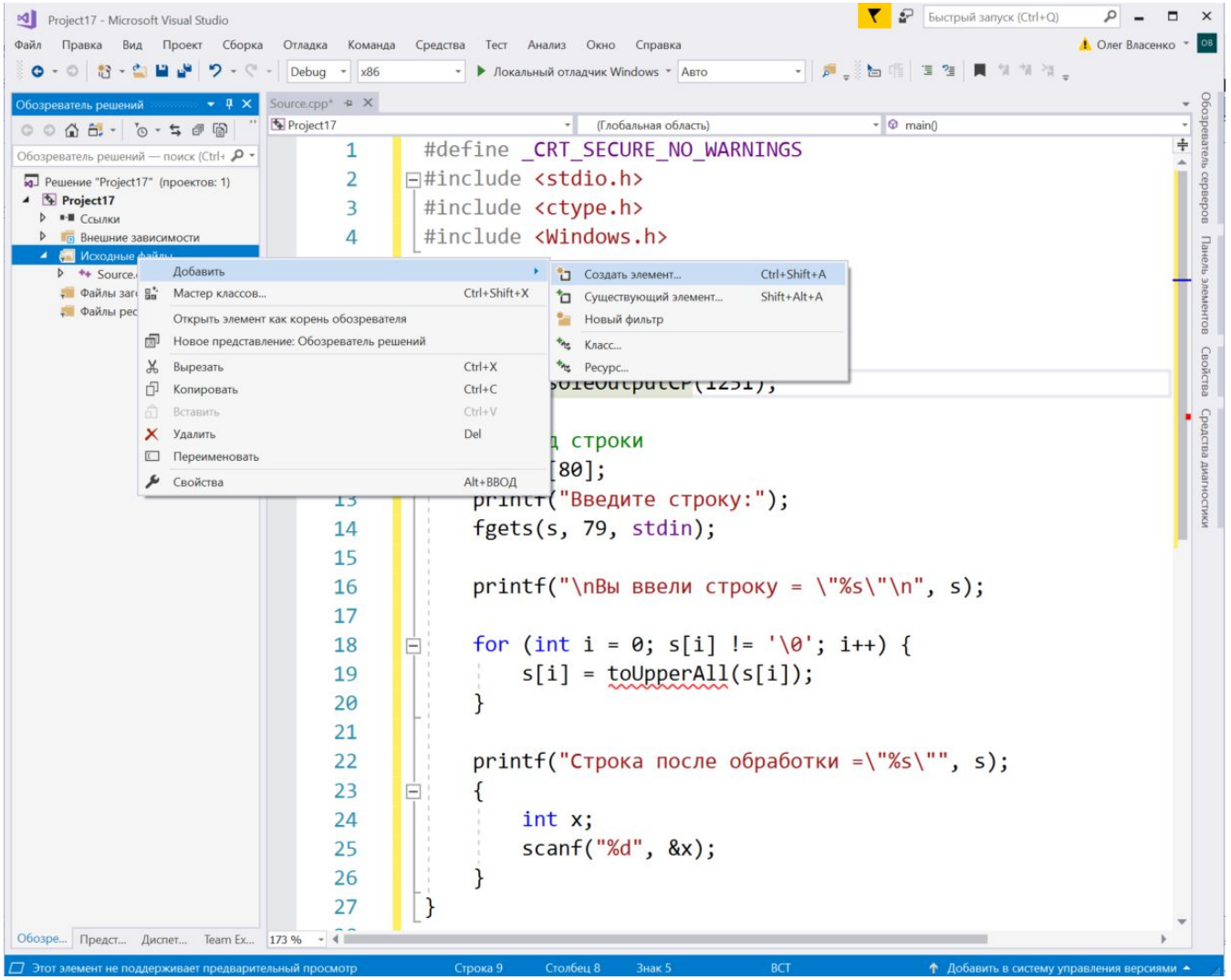
Задача 8* (3)

В главный модуль вставляем код main() с реализацией задачи 7.2



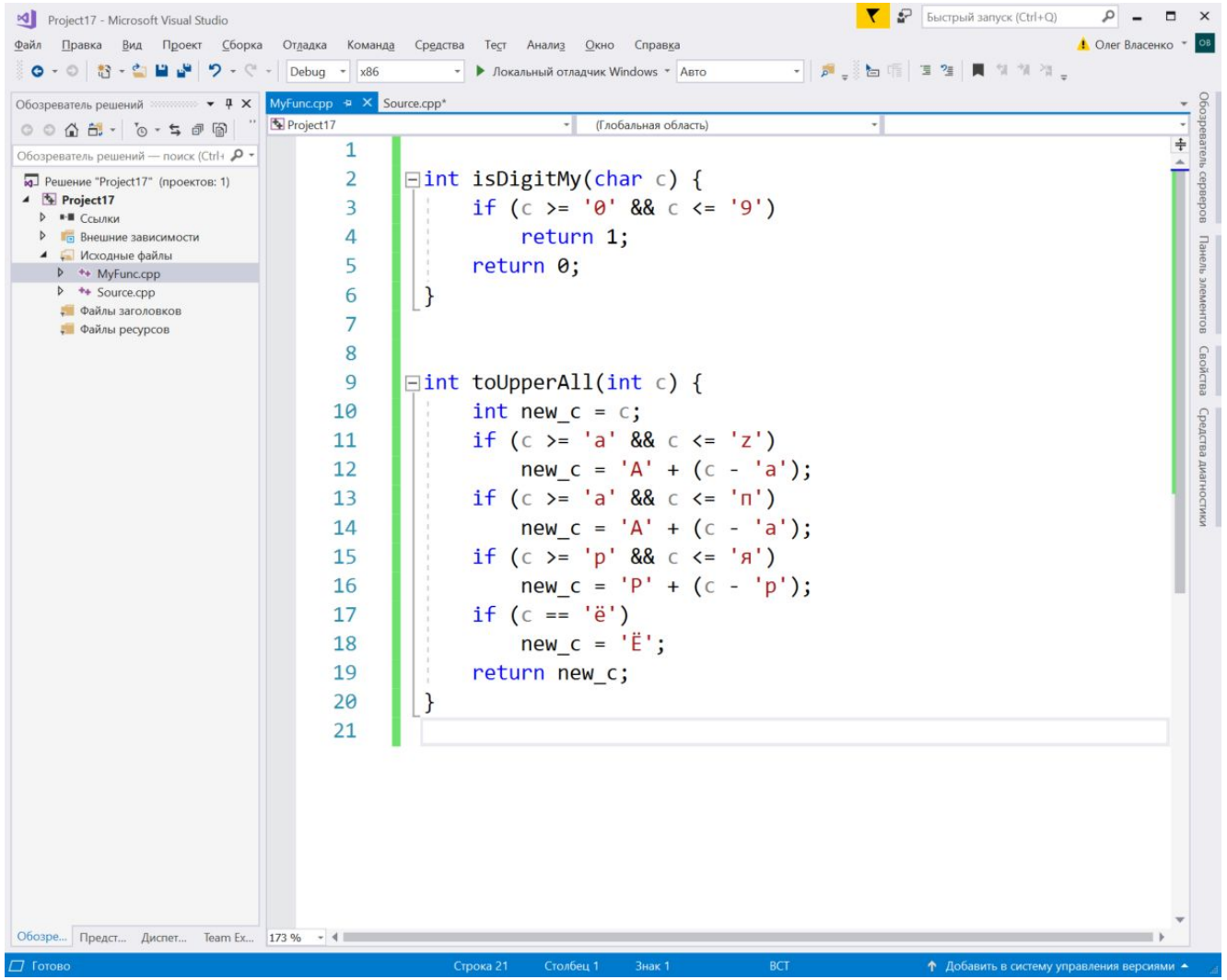
Задача 8* (4)

Создаем второй модуль – в нем будут находиться наши собственные реализации функций isDigitMy, toUpperAll и других.



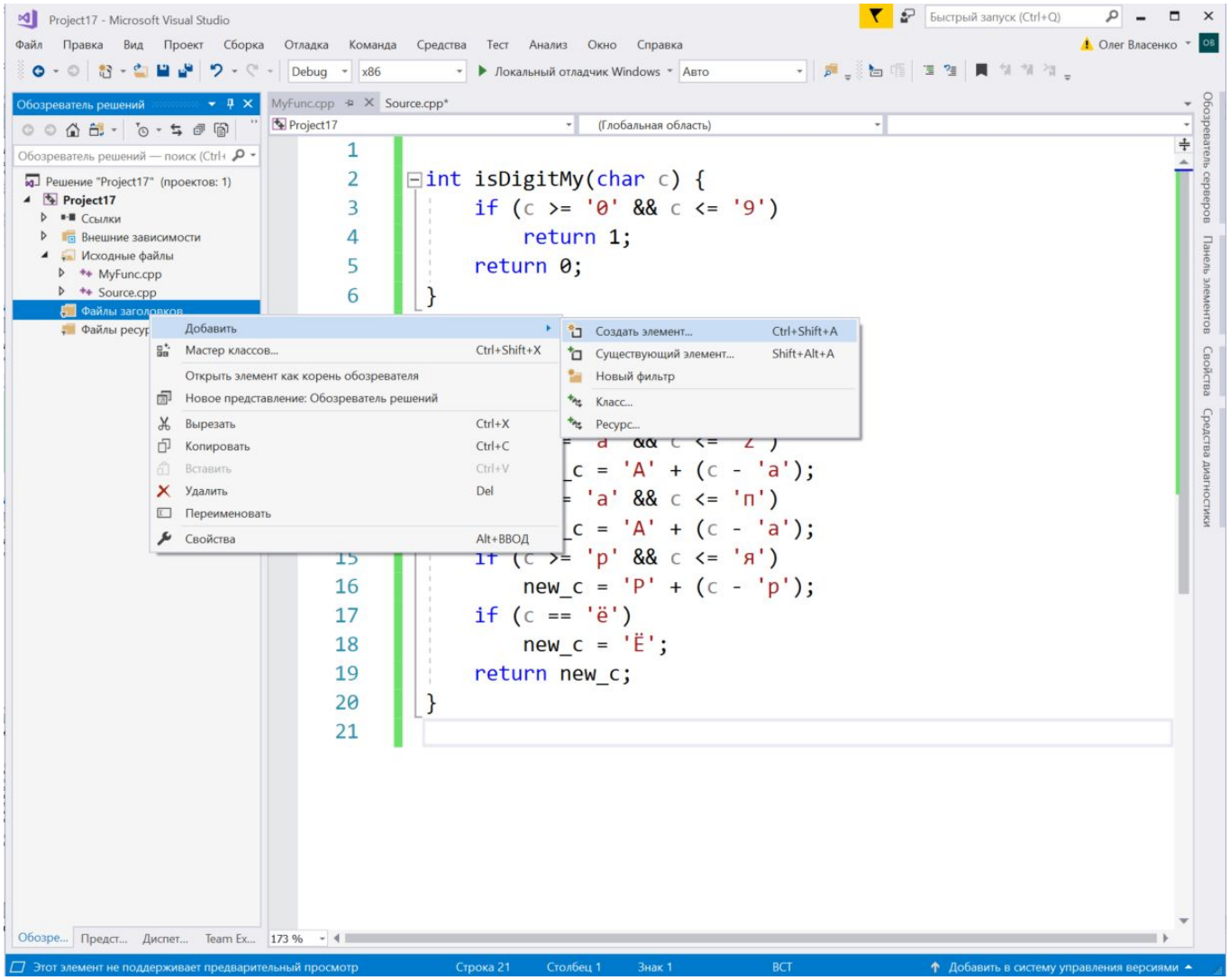
Задача 8* (5)

В CPP файл второго модуля вставляем код функций isDigitMy, toUpperAll



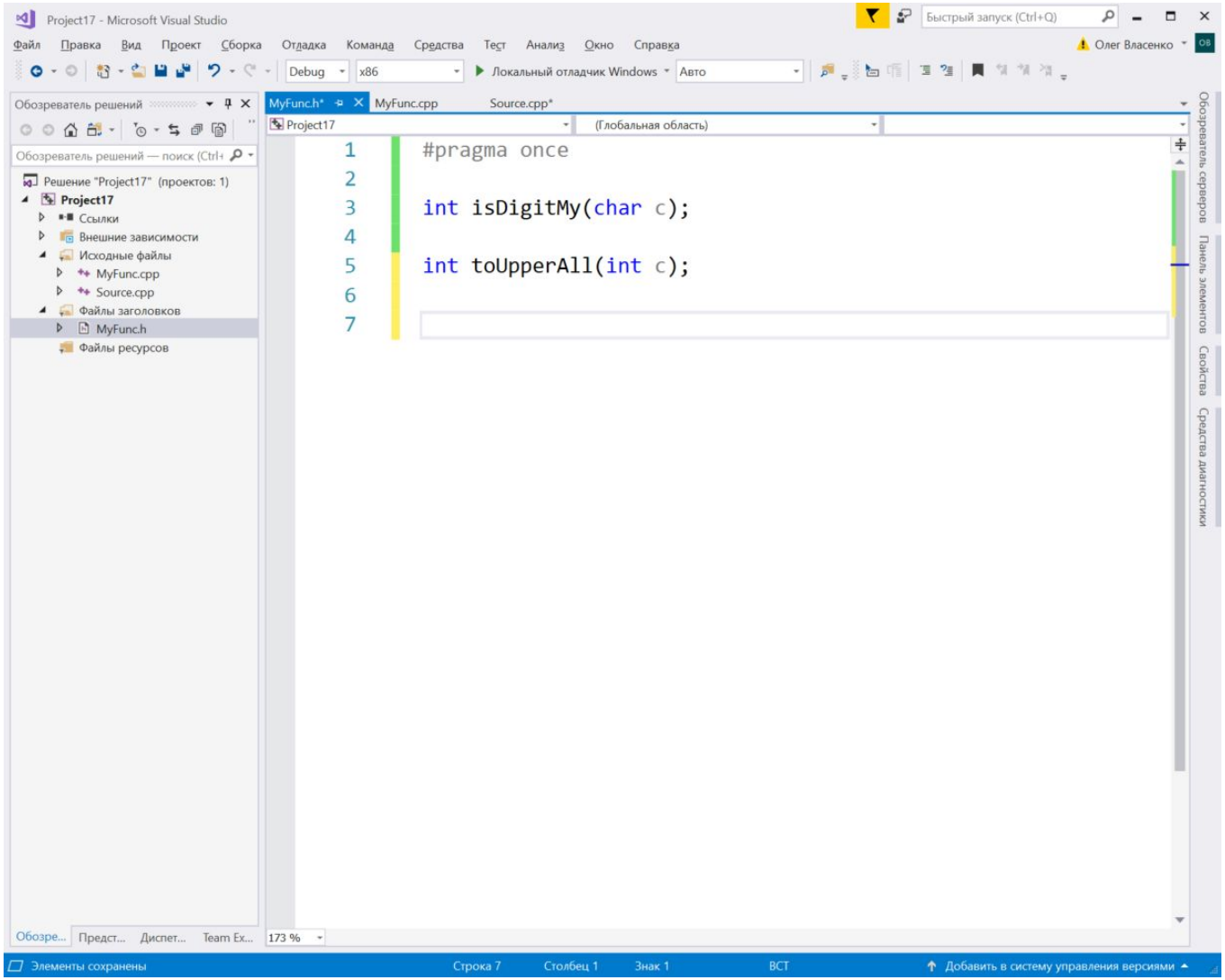
Задача 8* (6)

Создаем заголовочный файл для второго модуля



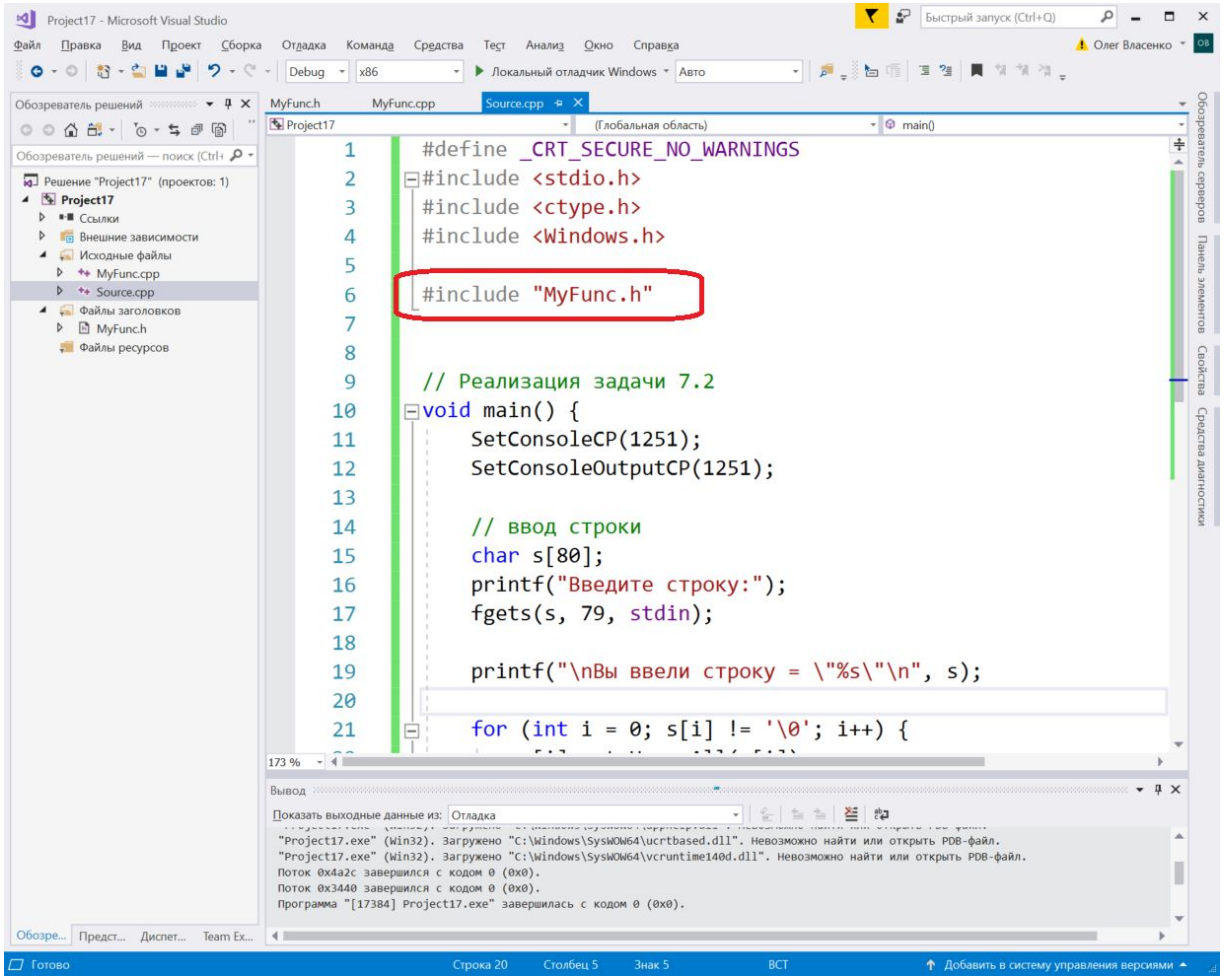
Задача 8* (7)

В заголовочный файл второго модуля добавляем прототипы функций isDigitMy, toUpperAll



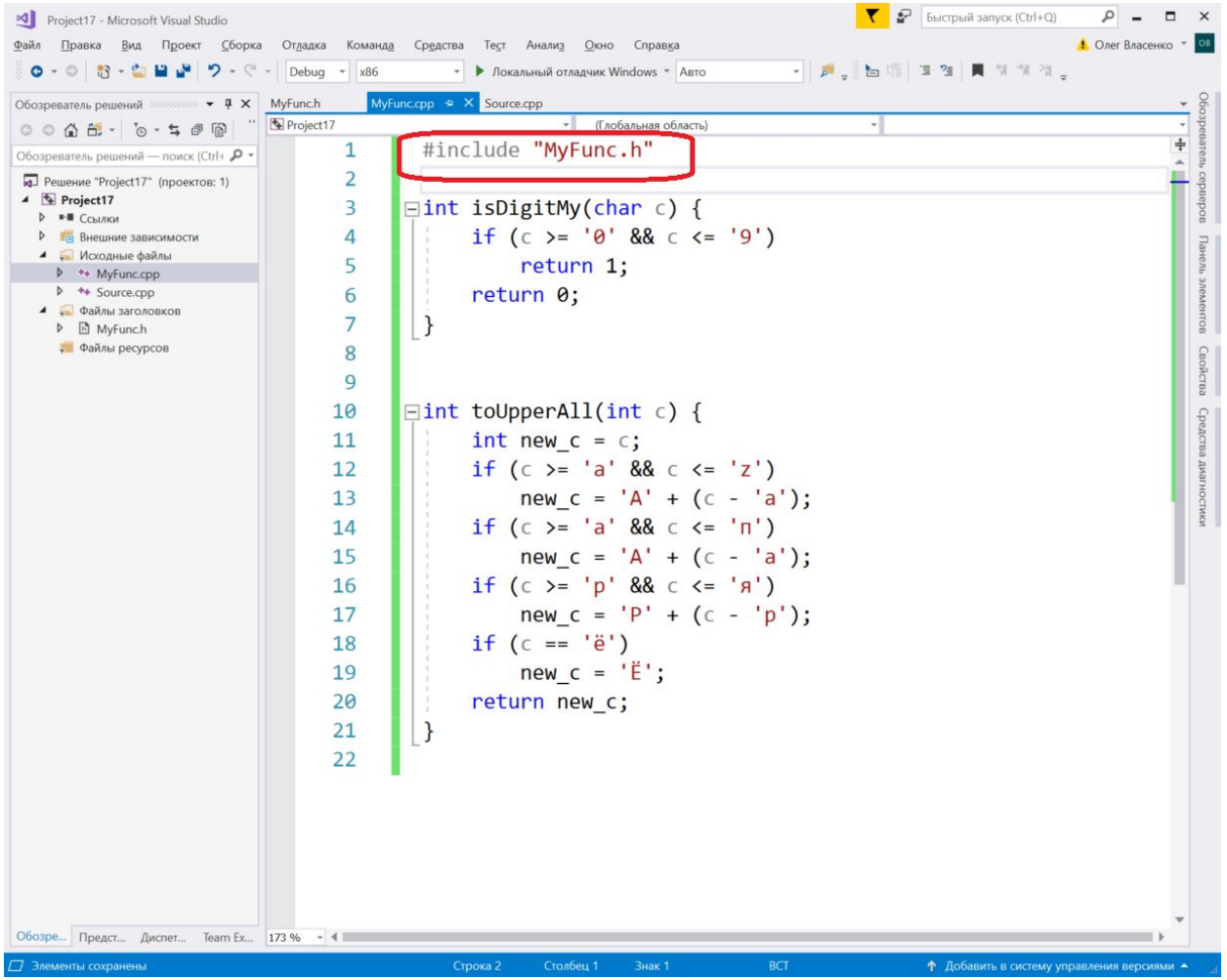
Задача 8* (8)

В главный модуль включаем заголовочный файл нашего второго модуля



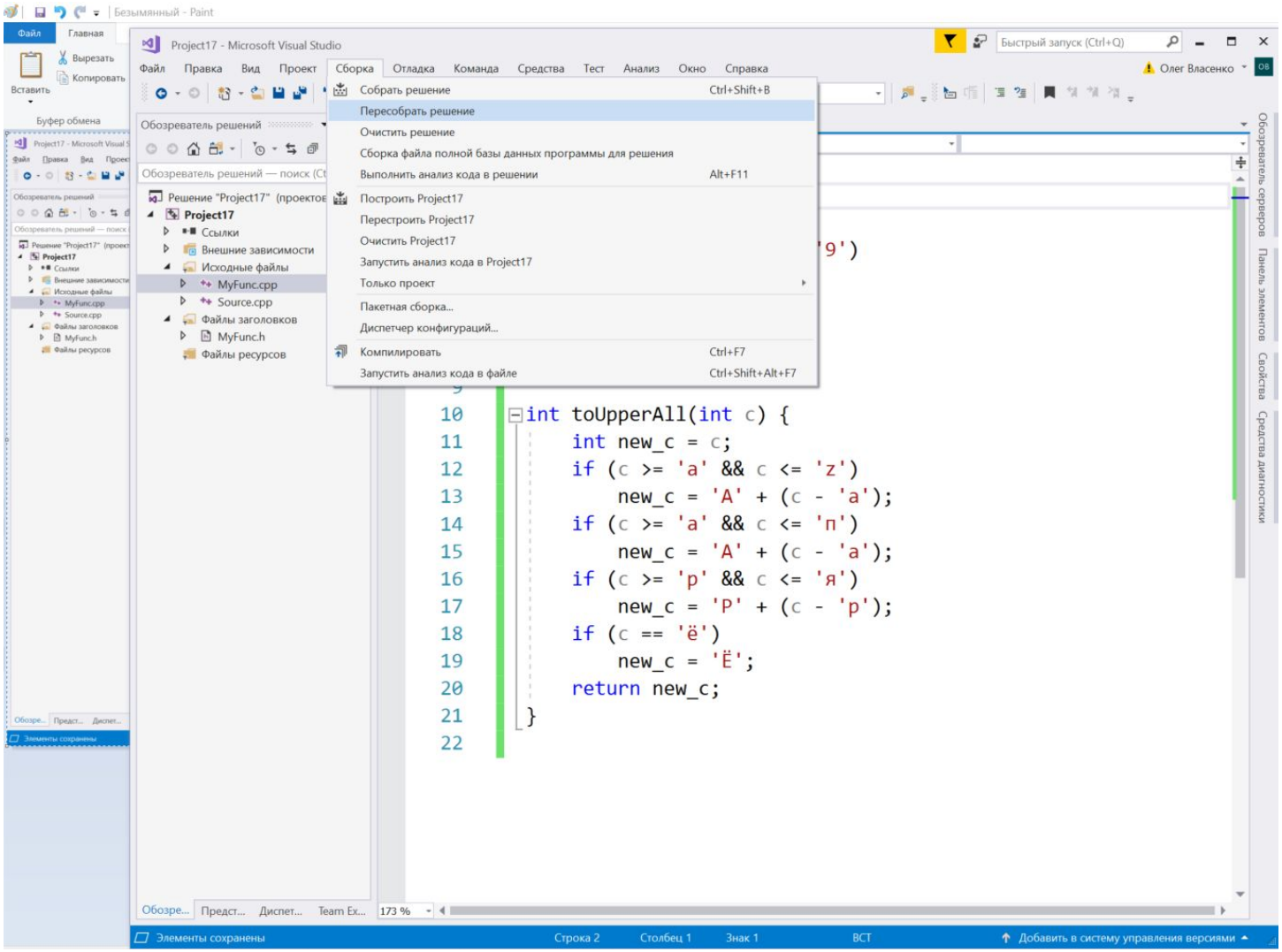
Задача 8* (9)

В CPP файл нашего второго модуля включаем заголовочный файл нашего же второго модуля



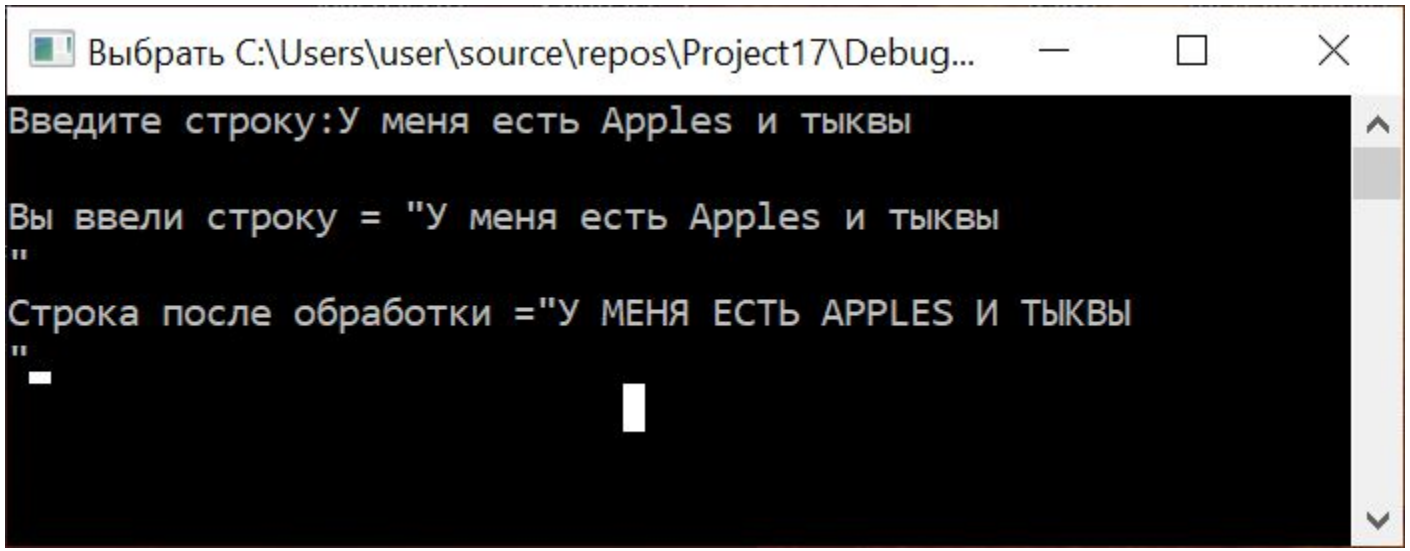
Задача 8* (10)

Собираем код



Задача 8* (11)

Запускаем на выполнение!



Задача 9 *

Реализовать собственную версию функции int strlen(char *)
Добавить ее во второй модуль
Написать код для проверки работоспособности этой функции – можно использовать код из лекции

Задача 10 **

Реализовать собственную версию функции int strcmp (char *, char *)
Добавить ее во второй модуль
Написать код для проверки работоспособности этой функции – можно использовать код из лекции

Задача 11 **

Реализовать собственную версию функции void strcpy (char *, char *)
Добавить ее во второй модуль
Написать код для проверки работоспособности этой функции – можно использовать код из лекции

Задача 12 **

Реализовать собственную версию функции void strcat (char *, char *)
Добавить ее во второй модуль
Написать код для проверки работоспособности этой функции – можно использовать код из лекции

Домашнее задание - часть 1

Доделать задачи с *, которые не были сделаны в классе.
Все собственные функции должны быть собраны во втором модуле.
В главном модуле для каждой из задач должен быть отдельный метод, в котором выполняет ввод, обработка и вывод.

** - если вы претендуете на уровень «Выше среднего» крайне полезно также доделать задачи с **

Домашнее задание - часть 2

Сделать индивидуальную задачу - выберите свой вариант преобразования – согласно номера в журнале!

Общее задание:
Нужно ввести одну строку с клавиатуры.
Введенную строку нужно вывести без изменений (для контроля) и вывести после обработки описанной в вашем варианте:

1. Все маленькие латинские буквы заменить символом ‘#’.
2. Все гласные латинские буквы заменить символом ‘&’.
3. Все согласные латинские буквы заменить символом ‘&’.
4. Все гласные большие латинские буквы заменить символом ‘$’.
5. Все гласные маленькие латинские буквы заменить символом ‘@’.
6. Все цифры заменить символом ‘X’.
7. Все латинские буквы сделать большими.
8. Все латинские буквы сделать маленькими.
9. Все символы за исключением латинских букв, заменить символом ‘_’ (подчеркивание).
10. Все знаки препинания заменить символом ‘_’ (подчеркивание).
11. Все символы арифметических операций, заменить символом ‘_’ (подчеркивание).

ИТОГО по лабораторной работе 20

1. Попробовали работать с символами
2. Попробовали работать со строками
3. Написали несколько собственных функций по обработке символов и строк
4. * Создали свой собственный модуль с созданными функциями обработки строк

Лабораторная работа №21

Изучение стандартной библиотеки Си. Модули обработки символов и строк

Задача 1 – Найти в интернете информацию

Нужно найти в интернете описание модулей ctype.h и string.h
Изучить поверхностно это описание.

Можно использовать например такие строки поиска:
«ctype.h язык Си»
«string.h язык Си»

Задача 2 – Выбрать функции для подробного изучения

Используя информацию из Интернета изучить список функций в модулях ctype.h и string.h.

Из модуля ctype.h взять функции isdigit, isupper, toupper
Из модуля ctype.h взять еще от 1 до 5 функций.

Из модуля string.h взять функции strlen, strcpy, strcmp.
Из модуля string.h взять еще от 1 до 5 функций.

Таким образом у каждого студента будет выбрано для изучения по 4-8 функций из каждого из модулей ctype.h и string.h

Список функций, которые нельзя выбирать – isalpha, strcat (они взяты в качестве образца ниже!)

Задача 3 – Подготовить код для одной из выбранных функций

Нужно подготовить код с комментариями, которого достаточно для того, чтобы вспомнить самому или объяснить другому как работает описываемая функция.

Структура:

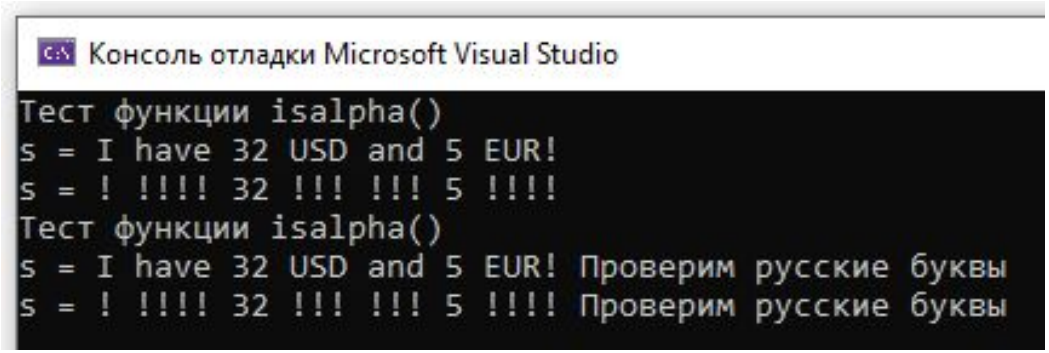
1) Источник информации (работающая ссылка)
2) Краткое описание - с указанием что делает функция, перечислением всех параметров и возвращаемого значения.
3) Код в котором есть образцы входных данных, есть пример использования функции, и есть вывод результата выполнения функции. Пример использования функции пояснен в коде - в виде комментариев.

Задача 3. Образец выполнения на примере isalpha() - 1

```
// Источник информации: https://ru.wikipedia.org/wiki/Ctype.h
// Краткое описание:
//  int isalpha(int c);
//  Возвращает ненулевое значение, если её аргумент является буквой,
//   в противном случае возвращается нуль.

void testIsAlpha1() {
    printf("Тест функции isalpha()\n");
    // Найти в строке все вхождения букв и заменить их на символ '!’
    char s[] = "I have 32 USD and 5 EUR! ";
    printf("s = %s\n", s);
    int i = 0;
    while (s[i] != '\0') {
        if (isalpha(s[i])) {
            s[i] = '!’;
        }
        i++;
    }
    printf("s = %s\n", s);
}
```

Задача 3. Образец выполнения на примере isalpha() - 2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <Windows.h>

void main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    testIsAlpha1();
    //testIsAlpha2();
    testIsAlpha3();
}
```



Задача 3. Образец выполнения на примере isalpha() - 3

```
void testIsAlpha2() {

    printf("Тест функции isalpha()\n");
    // Найти в строке все вхождения букв и заменить их на символ '!’
    char s[] = "I have 32 USD and 5 EUR! Привет!";
    printf("s = %s\n", s);
    int i = 0;
    while (s[i] != '\0') {
        if (isalpha(s[i])) {
            s[i] = '!’;
        }
        i++;
    }
    printf("s = %s\n", s);

}
```

Задача 3. Образец выполнения на примере isalpha() - 4

```
void testIsAlpha3() {

    printf("Тест функции isalpha()\n");
    // Найти в строке все вхождения букв и заменить их на символ '!’
    char s[] = "I have 32 USD and 5 EUR! Проверим русские буквы";
    printf("s = %s\n", s);
    int i = 0;
    while (s[i] != '\0') {
        if (isalpha((unsigned char)s[i])) {
            s[i] = '!’;
        }
        i++;
    }
    printf("s = %s\n", s);

}
```

Задача 4-10+ Подготовить код для каждой из выбранных функций

Как и в задаче 3 нужно подготовить код с комментариями, которого достаточно для того, чтобы вспомнить самому или объяснить другому как работает описываемая функция.

Структура:

1) Источник информации (работающая ссылка)
2) Краткое описание - с указанием что делает функция, перечислением всех параметров и возвращаемого значения.
3) Код в котором есть образцы входных данных, есть пример использования функции, и есть вывод результата выполнения функции. Пример использования функции пояснен в коде - в виде комментариев.

В качестве образца можно использовать код из задачи 3.
Примеры кода можно брать откуда угодно – из интернета, из книг, из примеров из лекции, придумывать самому, просить подсказать друзей.

Задача 4. Образец выполнения на примере strcat() - 1

```
// Источник информации: http://all-ht.ru/inf/prog/c/func/strcat.html
// Краткое описание:
//  char *strcat (char *destination, const char *append);
//  Аргументы:
//     destination – указатель на массив в который будет добавлена строка.
//     append – указатель на массив из которого будет скопирована строка.
//
//  Возвращаемое значение :
//     Функция возвращает указатель на массив, в который добавлена строка
(destination).
//
//  Описание :
//    Функция strcat добавляет в строку, на которую указывает аргумент destination,
//    строку, на которую указывает аргумент append.
//    Символ конца строки помещается в конце объединенных строк.
//    Если строки перекрываются, результат объединения будет не определен.
```

Задача 4. Образец выполнения на примере strcat() - 2

```
void testStrCat() {

    printf("Тест функции strcat()\n");

    char src[] = "Button";
    char dest[10] = "<>";
    printf("src = %s, dest = %s\n", src, dest);

    strcat(dest, src);
    printf("src = %s, dest = %s\n", src, dest);

    strcat(dest, "!");
    printf("src = %s, dest = %s\n", src, dest);
}


void main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    testStrCat();
}
```

ИТОГО по лабораторной работе 21

1. Попробовали самостоятельно найти информацию по стандартным функциям.
2. Потренировали создавать образцы использования стандартных функций.

ИТОГО по лекции 11

1. ASCII !!!
2. ASCIIZ !!!!!!!!
3. Вспомнили про указатели.
4. Узнали про операции над указателями (не только можно присваивать, получать адрес и обращаться к значению, но и складывать и вычитать!)
5. Узнали про связь указателей и массивов
6. Познакомились с десятком стандартных функций.
7. Получили задание и разобрали что делать на ЛР20 и ЛР21!