Лекция 03

БГТУ, ФИТ, ПОИТ, 2 семестр, Языки программирования

Структура языка программирования

1. **Спецификация системы программирования:**
   набор требований к системе программирования, достаточный для ее разработки.
   Система программирования – инструментальное ПО, предназначенное для разработки программного продукта на этапах программирования и отладки.

2. **Алфавит языка:**
   набор символов, разрешенных к использованию языком. Основывается на одной из кодировок.

   Совокупность символов, используемых в языке, называется алфавитом языка.
   Базовый набор символов исходного кода:
   - 52 символа латинского алфавит в нижнем и верхнем регистре;
   - Цифры от 0 до 9;
   - знаки операций;
   - символ подчеркивания;
   - 5 пробельных символов (пробел, символы горизонтальной и вертикальной табуляции, символ новой строки и символ перевода страницы);
   - ограничители и разделители;
   - специальные символы.

3. **Компилятор:**
   символы времени трансляции, символы времени выполнения.

   Текст программы на языке программирования хранится в исходных файлах и основан на определенной кодировке символов.

   Стандарт C++: исходный код ASCII.

   Набор символов времени выполнения: символы, интерпретируемые в среде выполнения. Любые дополнительные символы зависят от локализации.

4. **Компилятор CL:**
   исходный код C++ на ASCII, Windows-1251;

   - буквы латинского алфавита: [a…z], [A…Z];
   - цифры [0…9];
   - спецсимволы:_{}[]()#<>:;%.?*+-/^&~!=,”’ @ $
   - пробельные символы: пробел, символы табуляции, символы перехода на новую строку.

   Дополнительные символы времени выполнения определяются **setlocale**.

   По умолчанию, локаль: SetLocale (LC_ALL, "C") устанавливает стандартный контекст C.

   Во время выполнения можно изменить или запросить кодовую страницу языкового стандарта, используя вызов setlocale.
   Директива #pragma позволяет указать целевой языковой стандарт во время компиляции. Это гарантирует, что строки с расширенными символами будут сохраняться в правильном формате.

5. **Идентификатор:**
   имя компонента программы (переменной, функции, метки, типа и пр.), составленное программистом по определенным правилам.
   В разных языках программирования разные правила составления идентификаторов.

   Ruby
   Идентификаторы начинаются с буквы или специального модификатора.
   Основные правила:
   - имена локальных переменных начинаются со строчной буквы или знака подчеркивания (alpha, _ident);
   - имена глобальных переменных начинаются со знака доллара ($beta);
   - имена переменных экземпляра (принадлежащих объекту) начинаются со знака «@» (@foobar);
   - имена переменных класса (принадлежащих классу) предваряются двумя знаками «@» (@@not_const);
   - имена констант начинаются с прописной буквы (K6chip);
   - в именах идентификаторов знак подчеркивания _ можно использовать наравне со строчными буквами ($not_const);
   - имена специальных переменных, начинающиеся со знака «$» ($beta).

MS **Transact-SQL** – имена переменных должны начинаться с символа **@**.

Python
Используются символы Unicode.
Идентификаторы начинаются с латинской буквы, любого регистра или символа подчёркивания, могут содержать цифры.
Не могут совпадать с ключевыми словами (их список можно узнать по import keyword; print(keyword.kwlist), нежелательно переопределять встроенные имена.
Имена, начинающиеся с символа подчёркивания, имеют специальное значение.
В **Python 3** — в идентификаторе допустимы символы любого алфавита в Юникоде, например, кириллицы.

6. **Идентификатор C++**:
   - идентификаторы должны начинаться с буквы или подчеркивания;
   - идентификатор не может совпадать с ключевыми словами C++ или с именами библиотечных функций;
   - идентификаторы могут состоять из любого количества символов, но компилятор **гарантирует**, что будет считать значащими
     - 31 первых символов идентификаторов, не имеющих внешней связи;
     - не более 6 значащих символов идентификаторов с внешней связью;
   - идентификаторы чувствительны к регистру.

7. **Зарезервированные идентификаторы:**
   идентификаторы, которые предварительно определены в системе программирования.

   Python
   имеет особое значение идентификатор «_» — используется для хранения результата последнего вычисления.

8. **Зарезервированные идентификаторы C++**:
   все имена с двумя подчеркиваниями считаются зарезервированными.
   Кроме того: **is**xxxx, **mem**xxxx, **str**xxxx, **t**oxxxx, **wcs**xxxx, **Ецифра**xxxx, **LC_X**xxx, **SIGX**xxx, **SIG_X**xxxx.

9. **Литерал:**

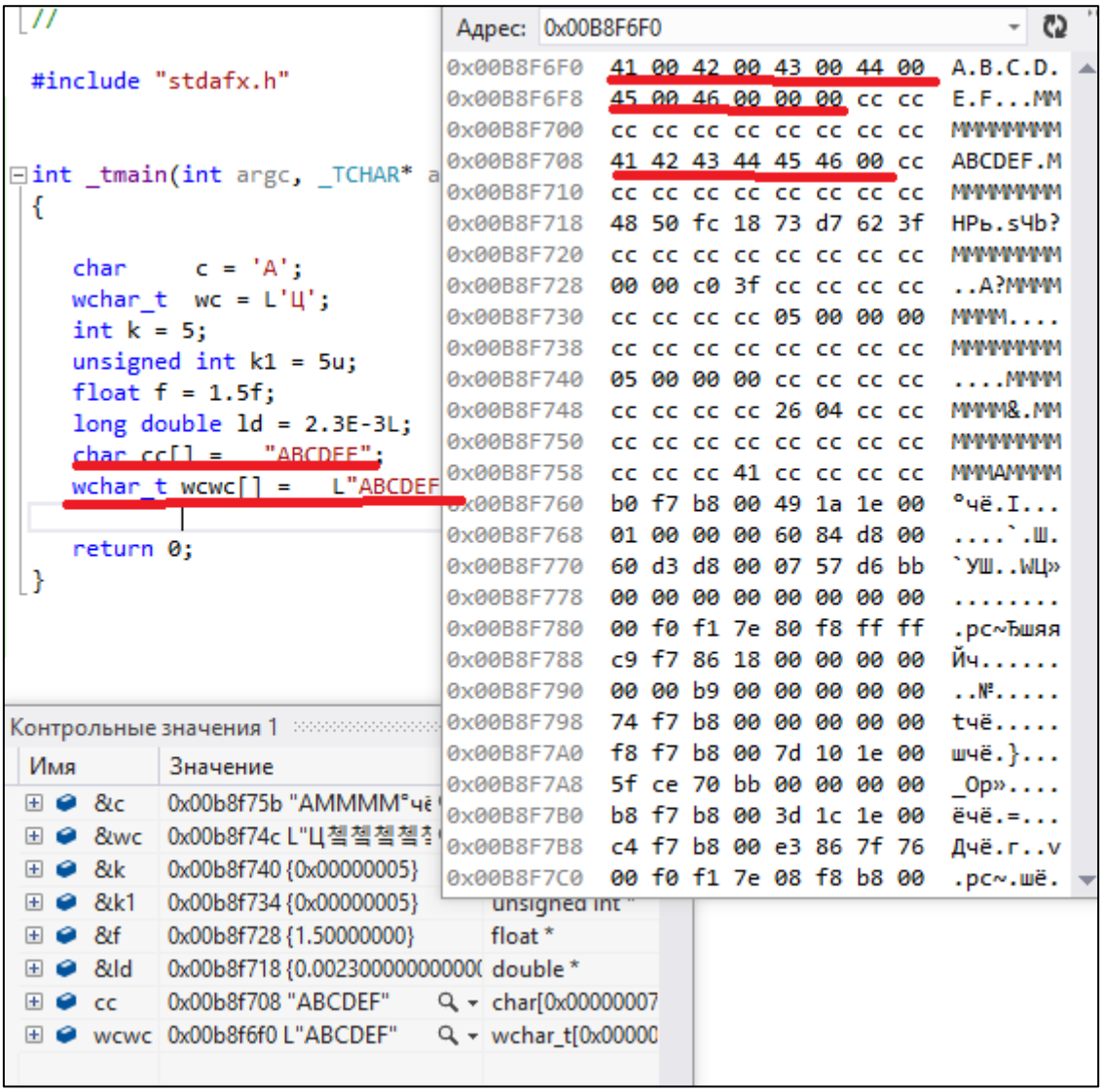
элемент программы, который непосредственно представляет значение.

В C++ существует четыре типа литералов:

- целочисленный литерал,
- вещественный литерал,
- символьный литерал,
- строковый литерал.

Управляющие символьные литералы:

\0	\x00	null	пустая литера
\a	\x07	bel	сигнал
\b	\x08	bs	возврат на шаг
\f	\x0C	ff	перевод страницы
\n	\x0A	lf	перевод строки
\r	\x0D	cr	возврат каретки
\t	\x09	ht	горизонтальная табуляция
\v	\x0B	vt	вертикальная табуляция
\\	\x5C	\	обратная косая черта
\'	\x27	'	
\"	\x22	"	
\?	\x3F	?	


//
#include "stdafx.h"
int _tmain(int argc, _TCHAR* a
{
char c = 'A';
wchar_t wc = L'Ц';
int k = 5;
unsigned int k1 = 5u;
float f = 1.5f;
long double ld = 2.3E-3L;
char cc[] = "ABCDEF";
wchar_t wcwc[] = L"ABCDEF
return 0;
}
Адрес: 0x00B8F6F0
0x00B8F6F0 41 00 42 00 43 00 44 00 A.B.C.D.
0x00B8F6F8 45 00 46 00 00 00 cc cc E.F...ММ
0x00B8F700 cc cc cc cc cc cc cc cc ММММММММ
0x00B8F708 41 42 43 44 45 46 00 cc ABCDEF.М
0x00B8F710 cc cc cc cc cc cc cc cc ММММММММ
0x00B8F718 48 50 fc 18 73 d7 62 3f HPь.sЧb?
0x00B8F720 cc cc cc cc cc cc cc cc ММММММММ
0x00B8F728 00 00 c0 3f cc cc cc cc ..А?ММММ
0x00B8F730 cc cc cc cc 05 00 00 00 ММММ....
0x00B8F738 cc cc cc cc cc cc cc cc ММММММММ
0x00B8F740 05 00 00 00 cc cc cc cc ....ММММ
0x00B8F748 cc cc cc cc 26 04 cc cc ММММ&.ММ
0x00B8F750 cc cc cc cc cc cc cc cc ММММММММ
0x00B8F758 cc cc cc 41 cc cc cc cc МММАММММ
0x00B8F760 b0 f7 b8 00 49 1a 1e 00 °чё.I...
0x00B8F768 01 00 00 00 60 84 d8 00 ....`.Ш.
0x00B8F770 60 d3 d8 00 07 57 d6 bb `УШ..WЦ»
0x00B8F778 00 00 00 00 00 00 00 00 ....
0x00B8F780 00 f0 f1 7e 80 f8 ff ff .рс~Ъшяя
0x00B8F788 c9 f7 86 18 00 00 00 00 Йч......
0x00B8F790 00 00 b9 00 00 00 00 00 ..№.....
0x00B8F798 74 f7 b8 00 00 00 00 00 tчё.....
0x00B8F7A0 f8 f7 b8 00 7d 10 1e 00 шчё.}...
0x00B8F7A8 5f ce 70 bb 00 00 00 00 _Ор»....
0x00B8F7B0 b8 f7 b8 00 3d 1c 1e 00 ёчё.=...
0x00B8F7B8 c4 f7 b8 00 e3 86 7f 76 Дчё.г..v
0x00B8F7C0 00 f0 f1 7e 08 f8 b8 00 .рс~.шё.
Контрольные значения 1
Имя
Значение
&c 0x00b8f75b "AMMMM°чё
&wc 0x00b8f74c L"Ц첼첼첼첼
&k 0x00b8f740 {0x00000005}
&k1 0x00b8f734 {0x00000005}
unsigned int
&f 0x00b8f728 {1.50000000}
float *
&ld 0x00b8f718 {0.002300000000000
double *
cc 0x00b8f708 "ABCDEF"
char[0x00000007
wcwc 0x00b8f6f0 L"ABCDEF"
wchar_t[0x00000

10. **Ключевые слова:**

последовательности символов алфавита языка, имеющие специальное назначение.

Ключевые слова зарезервированы компилятором для обозначения типов переменных, класса хранения, элементов операторов.

Ruby:

BEGIN	END	alias	and	begin
break	case	class	def	defined?
do	else	elsif	end	ensure
false	**for**	**if**	in	module
next	nil	not	or	redo
rescue	retry	return	self	super
then	true	undef	unless	until
when	while	yield		

Python (не могут быть использованы как обычные идентификаторы)

false	class	finally	is	return
none	continue	**for**	lambda	try
true	def	from	nonlocal	while
and	del	global	not	with
as	elif	**if**	or	yield
assert	else	import	pass	print
break	except	in	raise	

Go

break	case	chan	const	continue	default	
defer	else	fallthrough	**for**	func	go	
goto	**if**	import	interface	map	package	
range	return	select	struct	switch	type	var

11. **Ключевые слова C++:**

https://docs.microsoft.com/ru-ru/cpp/cpp/keywords-cpp?view=vs-2019

Примеры ключевых слов C++

break	case	catch	char
char16_t	char32_t	class	const
false	finally	float	for
inline	return	**if**	struct
__cdecl	__int16 4	__int32 4	__int64

12. **Типы данных:**
    - фундаментальные (или встроенные);
    - определенные программистом.

Тип данных определяет:

- внутреннее представление данных в памяти компьютера;
- диапазон значений, которые могут принимать величины этого типа;
- операции и функции, которые можно применять к величинам этого типа.

В **Java**: 8 примитивных типов: boolean, byte, char, short, int, long, float, double.

Длины и диапазоны значений примитивных типов определяются **стандартом**, а не реализацией:

Тип	Длина (в байтах)	Диапазон или набор значений
boolean	1 в массивах, 4 в переменных	true, false
byte	1	−128..127
char	2	$0..2^{16}-1$, или 0..65535
short	2	$-2^{15}..2^{15}-1$, или −32768..32767
int	4	$-2^{31}..2^{31}-1$, или −2147483648..2147483647
long	8	$-2^{63}..2^{63}-1$, или примерно $-9.2\cdot10^{18}..9.2\cdot10^{18}$
float	4	$-(2-2^{-23})\cdot2^{127}..(2-2^{-23})\cdot2^{127}$, или примерно $-3.4\cdot10^{38}..3.4\cdot10^{38}$, а также NaN
double	8	$-(2-2^{-52})\cdot2^{1023}..(2-2^{-52})\cdot2^{1023}$, или примерно $-1.8\cdot10^{308}..1.8\cdot10^{308}$, а также NaN

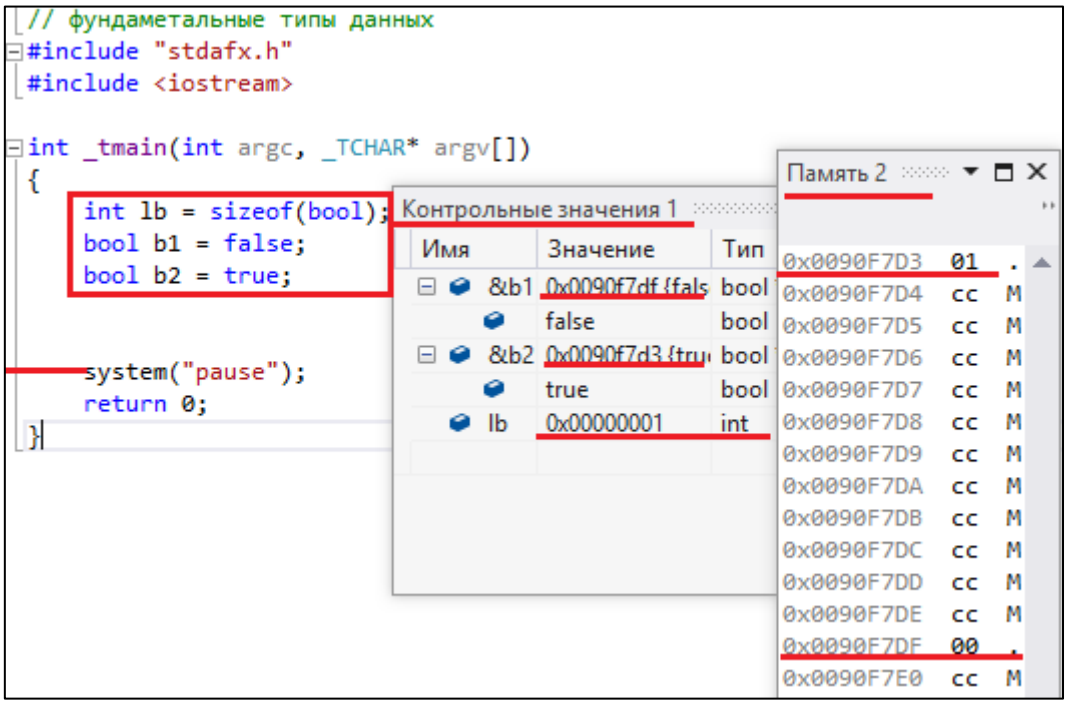
Для описания основных типов C++ определены следующие ***ключевые*** слова:
- int (целый);
- char (символьный);
- wchar_t (расширенный символьный);
- bool (логический);
- float (вещественный);
- double (вещественный с двойной точностью);
- тип void.

Модификаторы основных типов, уточняющие внутреннее представление и диапазон значений стандартных типов:
- short (короткий);
- long (длинный);
- signed (знаковый);
- unsigned (беззнаковый).

Размеры основных типов

Тип	Размер
bool, char, unsigned char, signed char, __int8	1 байт
__int16, short, unsigned short, wchar_t, __wchar_t	2 байта
float, __int32, int, unsigned int, long, unsigned long	4 байта
double, __int64, long double, long long	8 байт
__int128	16 байт

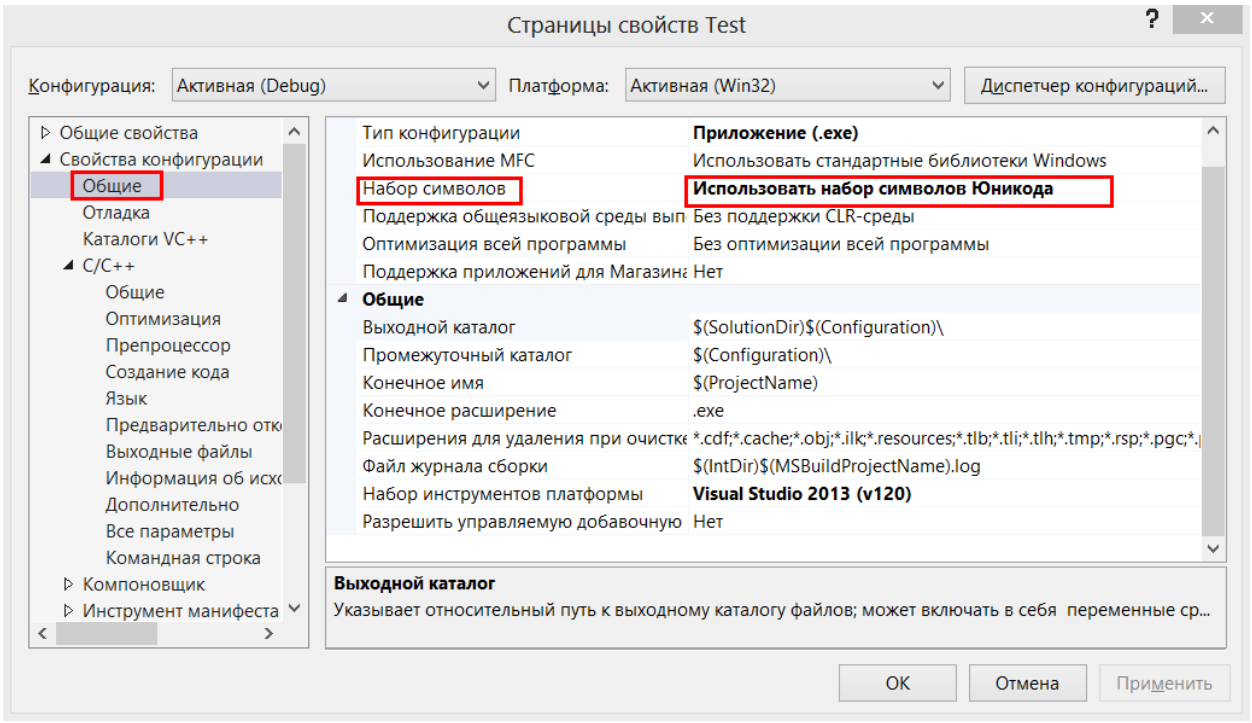
13. **Фундаментальные типы C++:**
bool

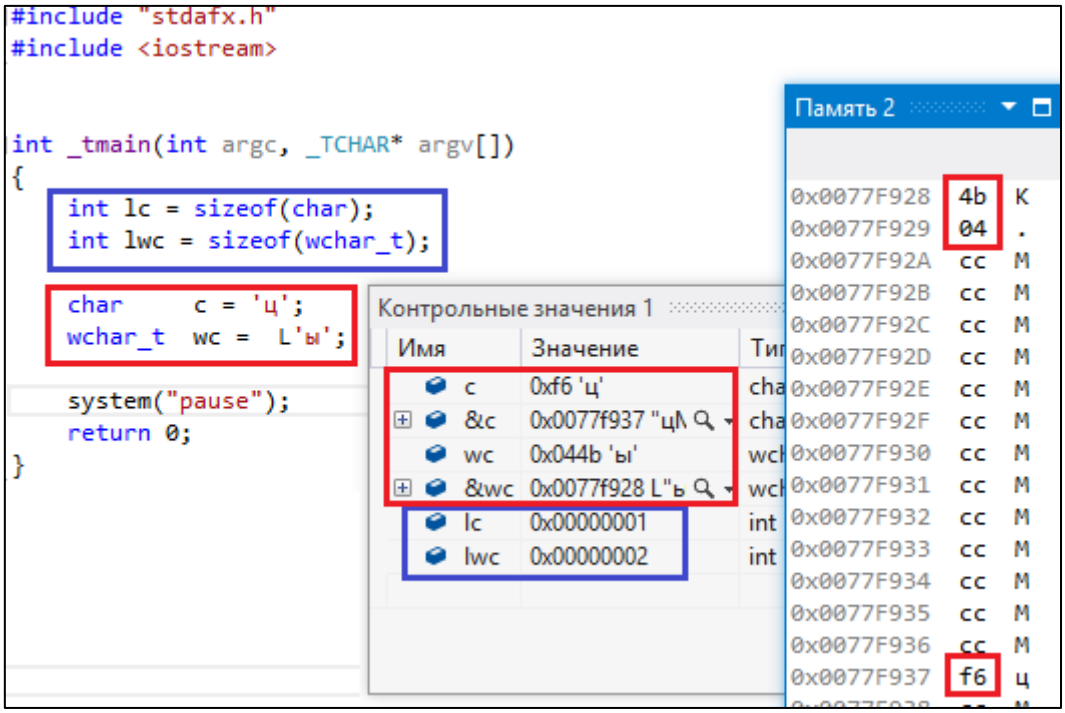


14. **Фундаментальные типы C++:**

char

wchar_t



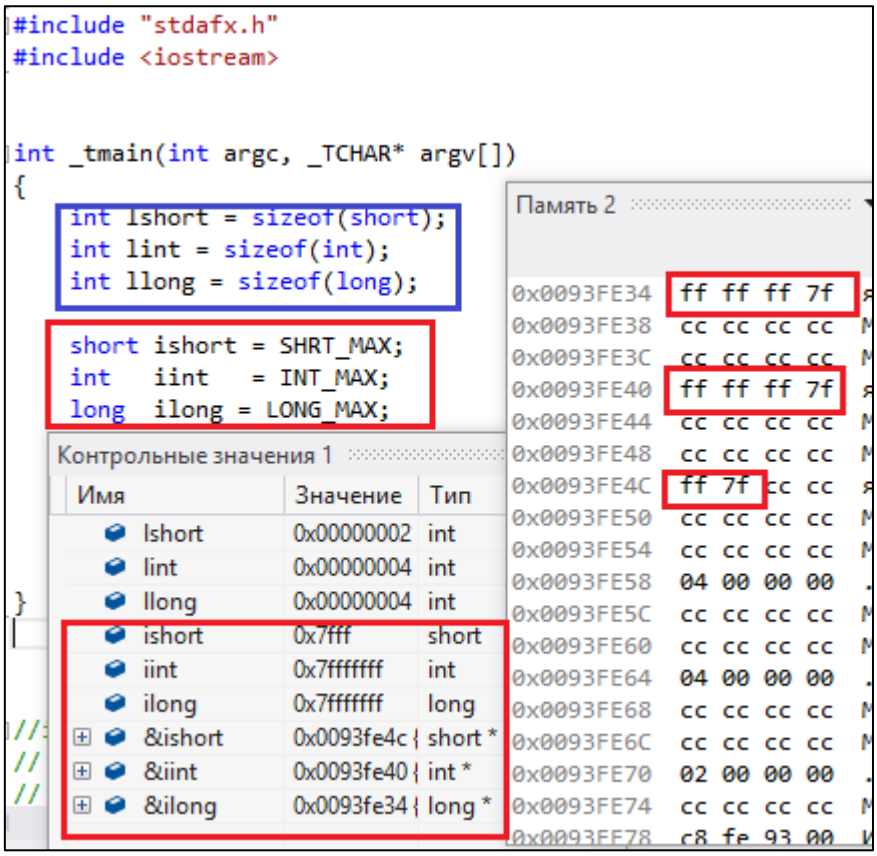


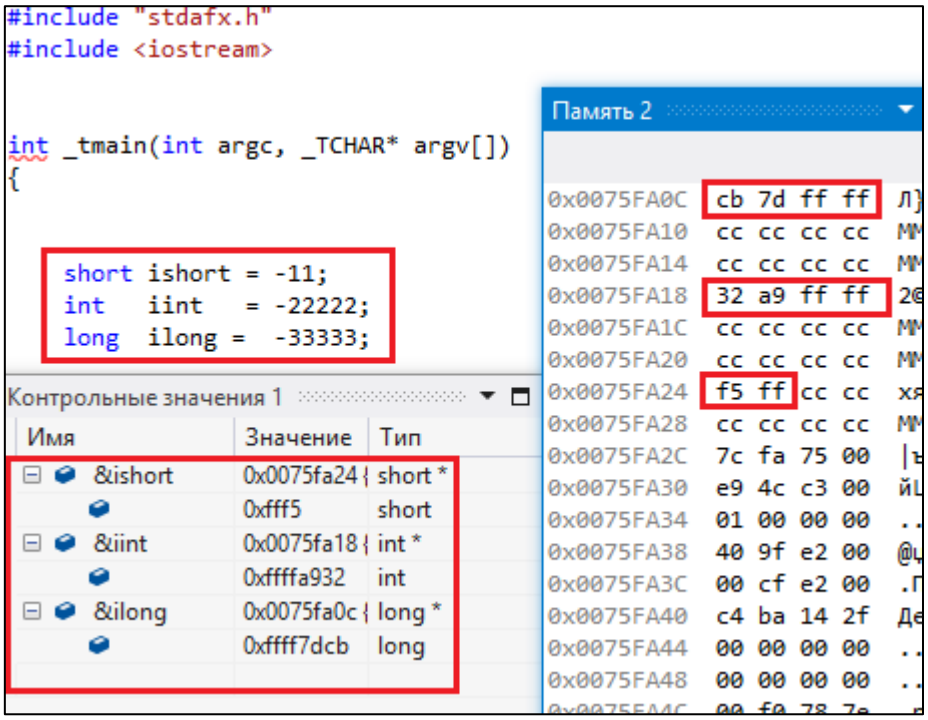
15. **Фундаментальные типы C++:**
[unsigned] int
[unsigned] short
[unsigned] long

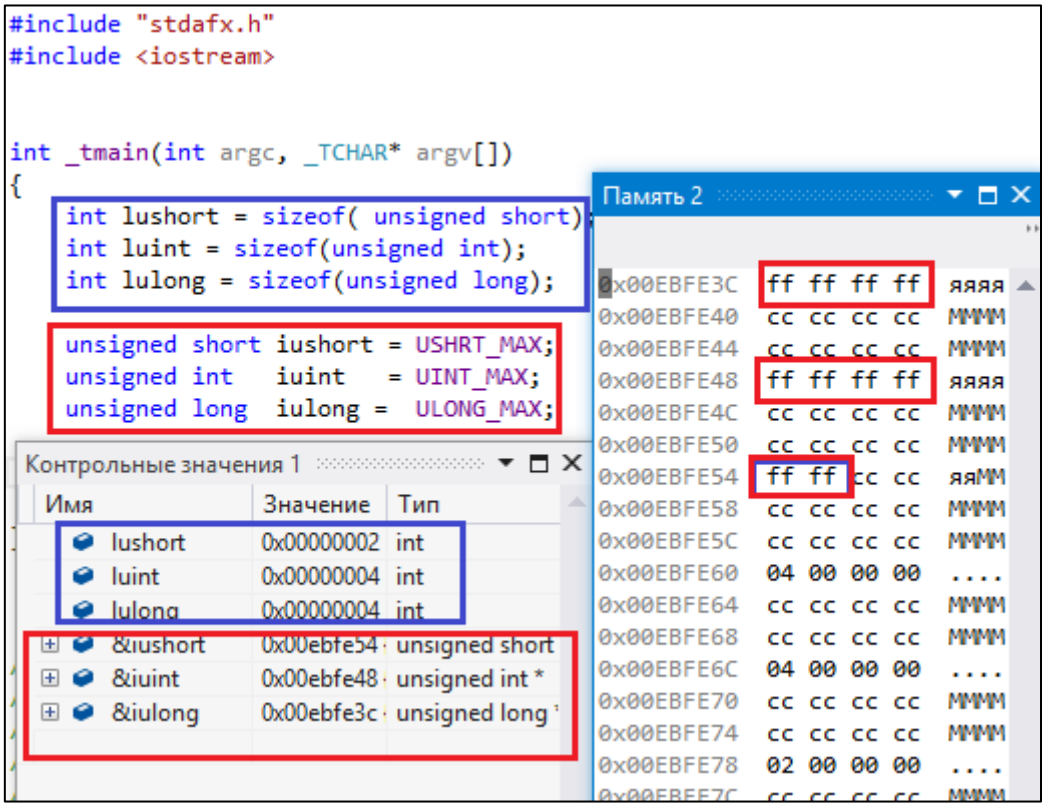
Внутреннее представление величины целого типа:

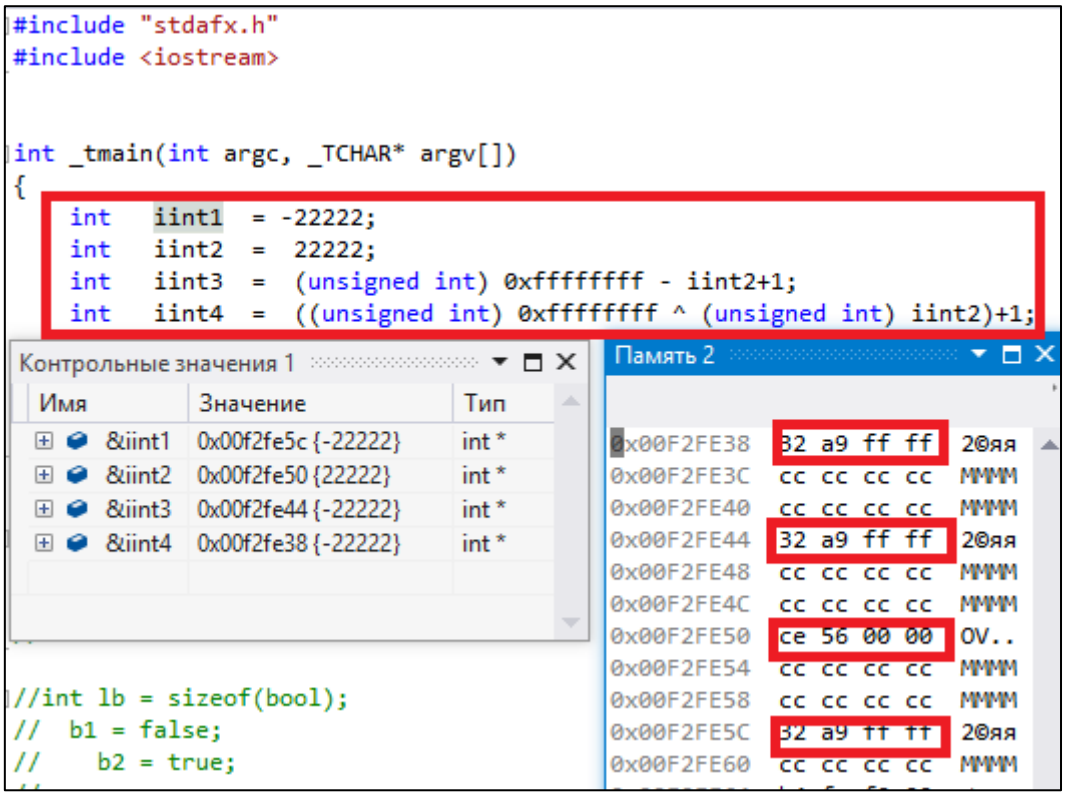
- целое число в двоичном коде.
- спецификатор **signed**: старший разряд (бит) числа интерпретируется как знаковый (0 — положительное число, 1 — отрицательное).
- спецификатор **unsigned**: старший разряд (бит) рассматривается как значащий, позволяет представлять только положительные числа.

По умолчанию все целочисленные типы считаются знаковыми, то есть спецификатор signed можно опускать.




#include "stdafx.h"
#include <iostream>
int _tmain(int argc, _TCHAR* argv[])
{
short ishort = -11;
int iint = -22222;
long ilong = -33333;
Память 2
0x0075FA0C cb 7d ff ff
0x0075FA10 cc cc cc cc
0x0075FA14 cc cc cc cc
0x0075FA18 32 a9 ff ff
0x0075FA1C cc cc cc cc
0x0075FA20 cc cc cc cc
0x0075FA24 f5 ff cc cc
0x0075FA28 cc cc cc cc
0x0075FA2C 7c fa 75 00
0x0075FA30 e9 4c c3 00
0x0075FA34 01 00 00 00
0x0075FA38 40 9f e2 00
0x0075FA3C 00 cf e2 00
0x0075FA40 c4 ba 14 2f
0x0075FA44 00 00 00 00
0x0075FA48 00 00 00 00
Контрольные значения 1
Имя
Значение
Тип
&ishort 0x0075fa24 short *
0xfff5 short
&iint 0x0075fa18 int *
0xffffa932 int
&ilong 0x0075fa0c long *
0xffff7dcb long


#include "stdafx.h"
#include <iostream>
int _tmain(int argc, _TCHAR* argv[])
{
int lushort = sizeof(unsigned short);
int luint = sizeof(unsigned int);
int lulong = sizeof(unsigned long);
unsigned short iushort = USHRT_MAX;
unsigned int iuint = UINT_MAX;
unsigned long iulong = ULONG_MAX;
Память 2
0x00EBFE3C ff ff ff ff яяяя
0x00EBFE40 cc cc cc cc MMMM
0x00EBFE44 cc cc cc cc MMMM
0x00EBFE48 ff ff ff ff яяяя
0x00EBFE4C cc cc cc cc MMMM
0x00EBFE50 cc cc cc cc MMMM
0x00EBFE54 ff ff cc cc яяMM
0x00EBFE58 cc cc cc cc MMMM
0x00EBFE5C cc cc cc cc MMMM
0x00EBFE60 04 00 00 00 ....
0x00EBFE64 cc cc cc cc MMMM
0x00EBFE68 cc cc cc cc MMMM
0x00EBFE6C 04 00 00 00 ....
0x00EBFE70 cc cc cc cc MMMM
0x00EBFE74 cc cc cc cc MMMM
0x00EBFE78 02 00 00 00 ....
Контрольные значения 1
Имя
Значение
Тип
lushort 0x00000002 int
luint 0x00000004 int
lulong 0x00000004 int
&iushort 0x00ebfe54 unsigned short
&iuint 0x00ebfe48 unsigned int *
&iulong 0x00ebfe3c unsigned long *



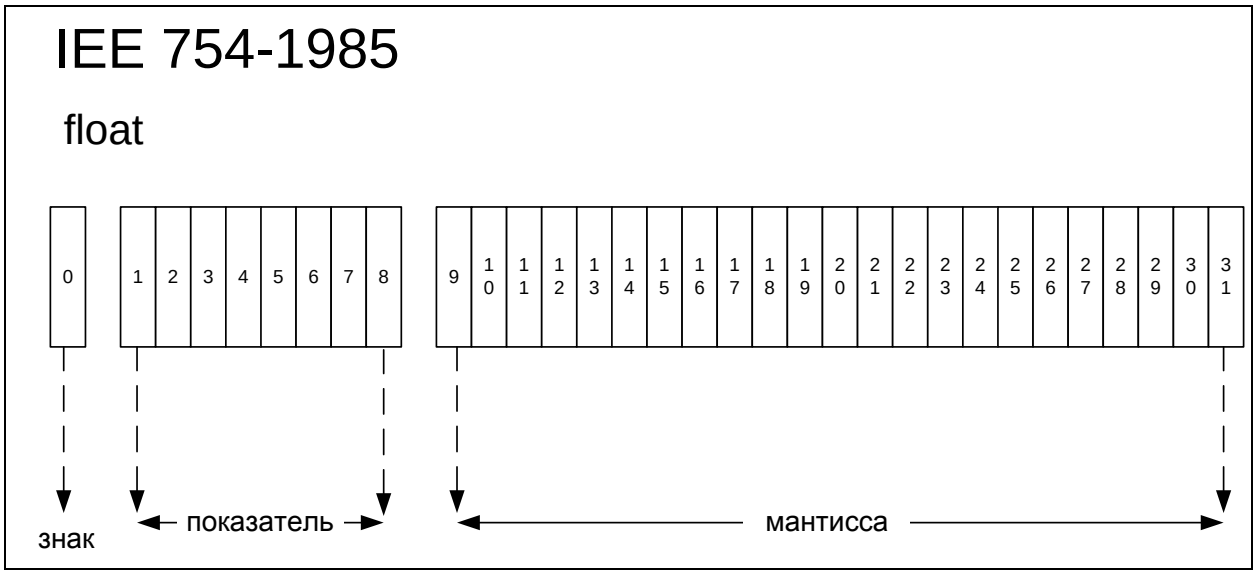
Два стандартных включаемых заголовочных файла, <limits.h> и <float.h>, определяют числовые ограничения или минимальное и максимальное значения, которые может хранить переменная данного типа.

Ограничения для некоторых целочисленных типов, заданные в стандартном файле заголовка <limits.h>, представлены в таблице:

Константа	**Значение**	**Значение**
CHAR_BIT	Количество битов в наименьшей переменной, которая не является битовым полем.	8
SCHAR_MIN	Минимальное значение для переменной типа **signed char**.	–128
SCHAR_MAX	Максимальное значение для переменной типа **signed char**.	127
UCHAR_MAX	Максимальное значение для переменной типа **unsigned char**.	255 (0xff)

16. **Фундаментальные типы C++:**
    float
    double

Стандарт C++ определяет три типа данных для хранения вещественных значений: float, double и long double.



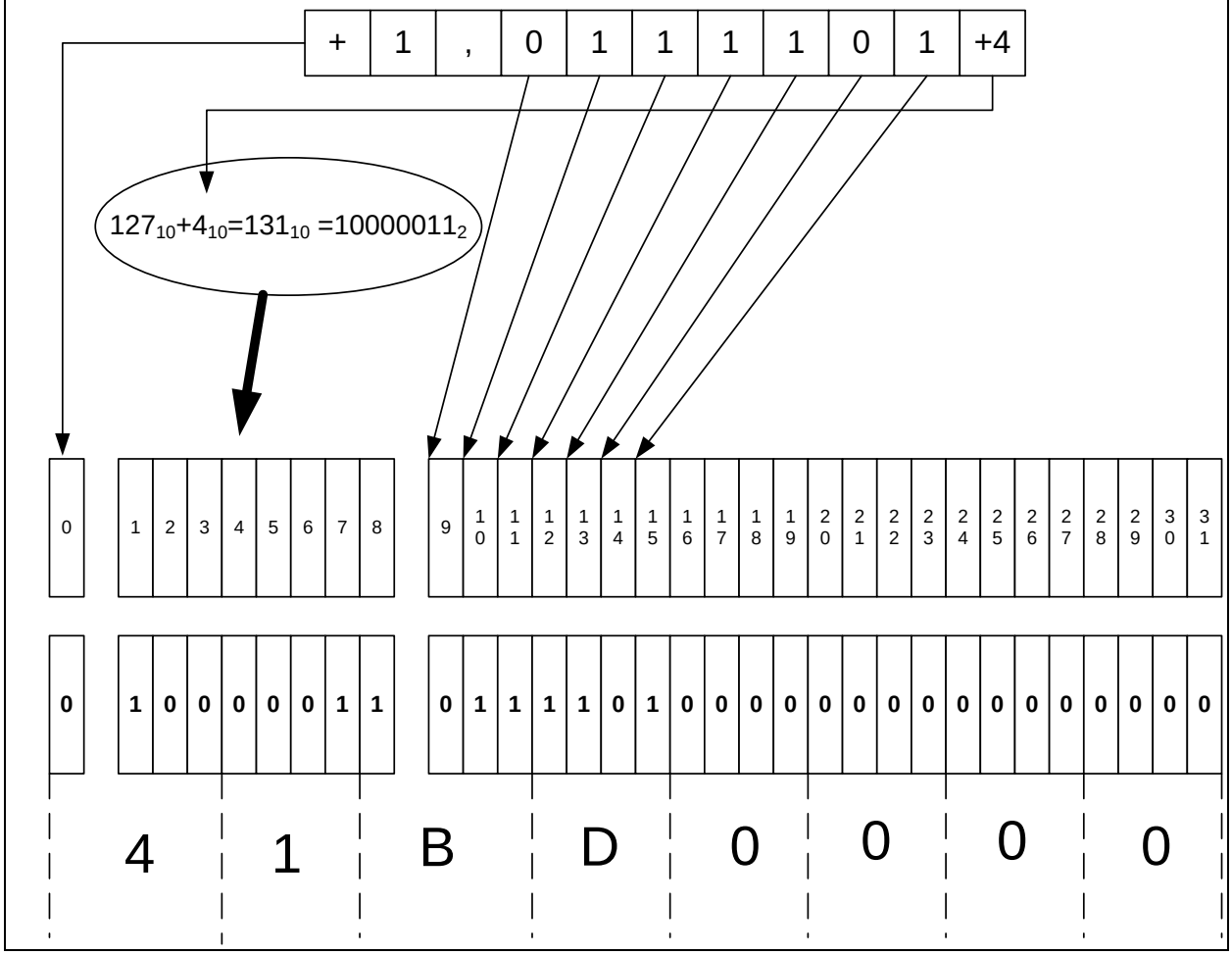
$23{,}625_{10} = 10111{,}101_2 = 1{,}0111101_2(+4)$

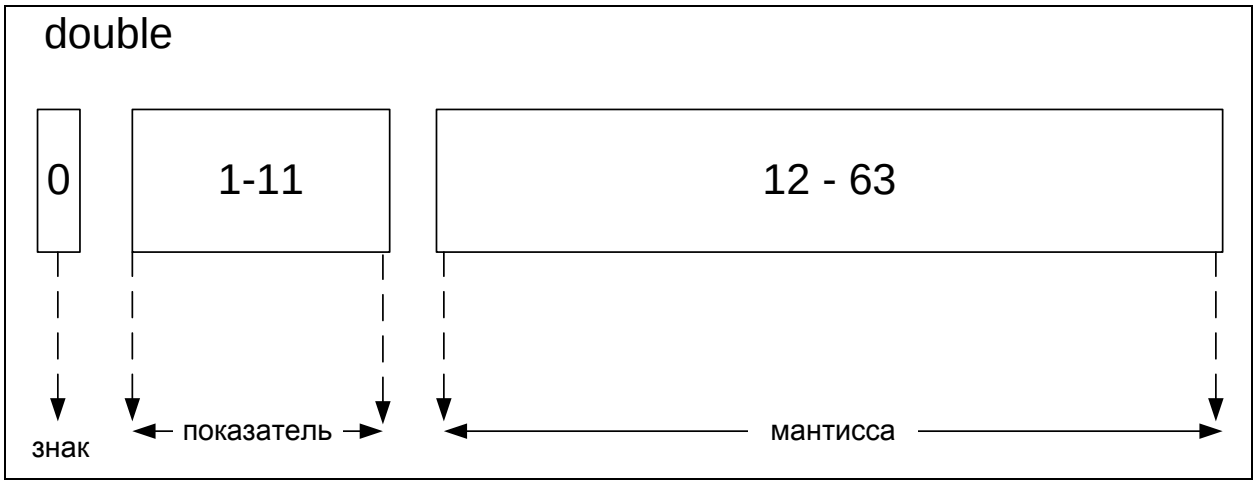
$23_{10} = 10111_2$

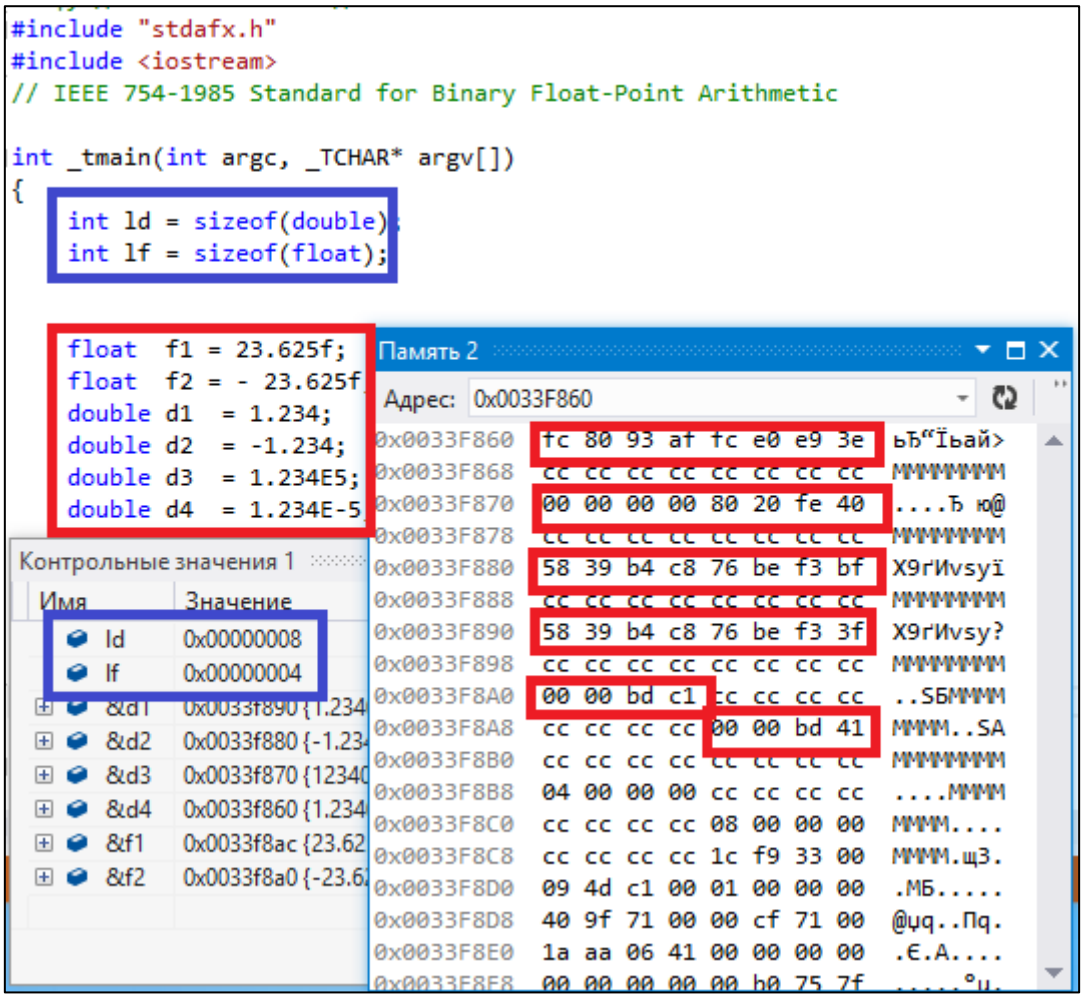
$0{,}625_{10} = 0{,}101$

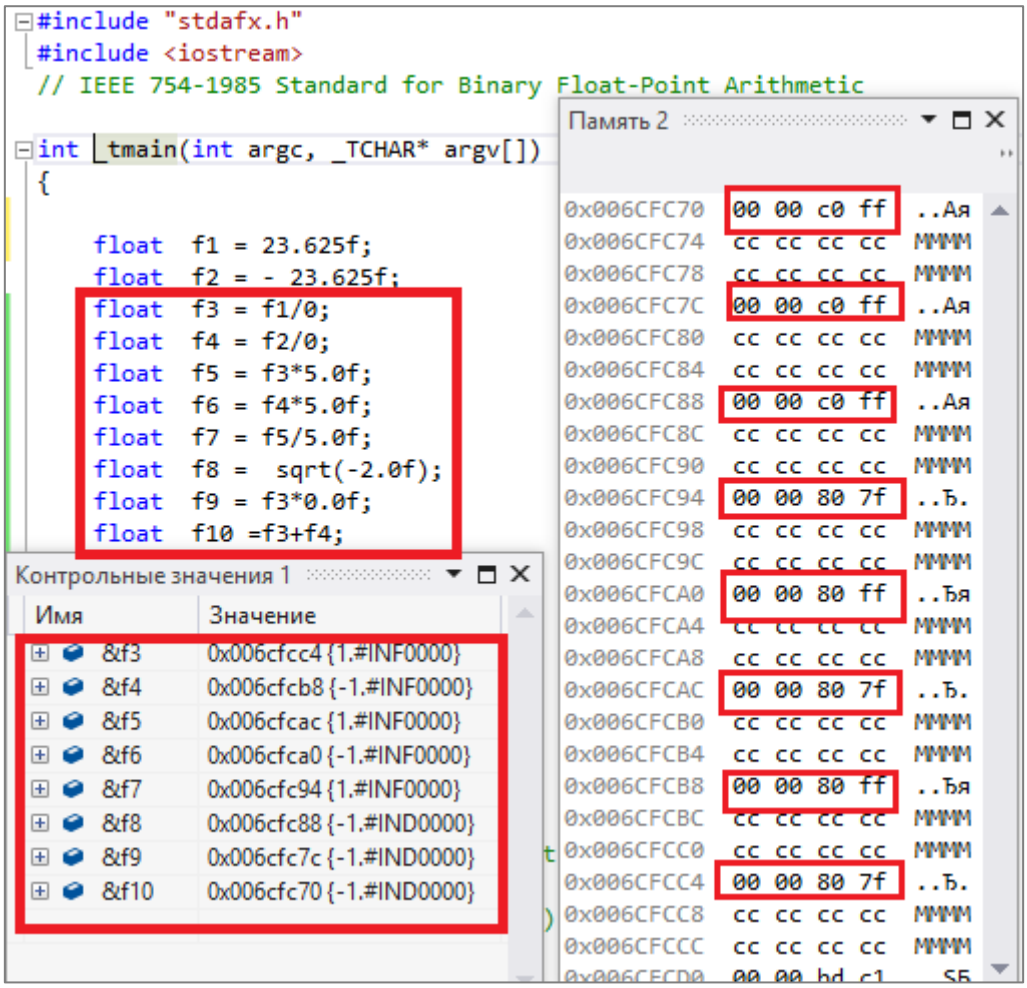
$$+23{,}625 = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$$

$$+23{,}625 = +10111{,}101 = +1{,}0111101\ (+4)$$



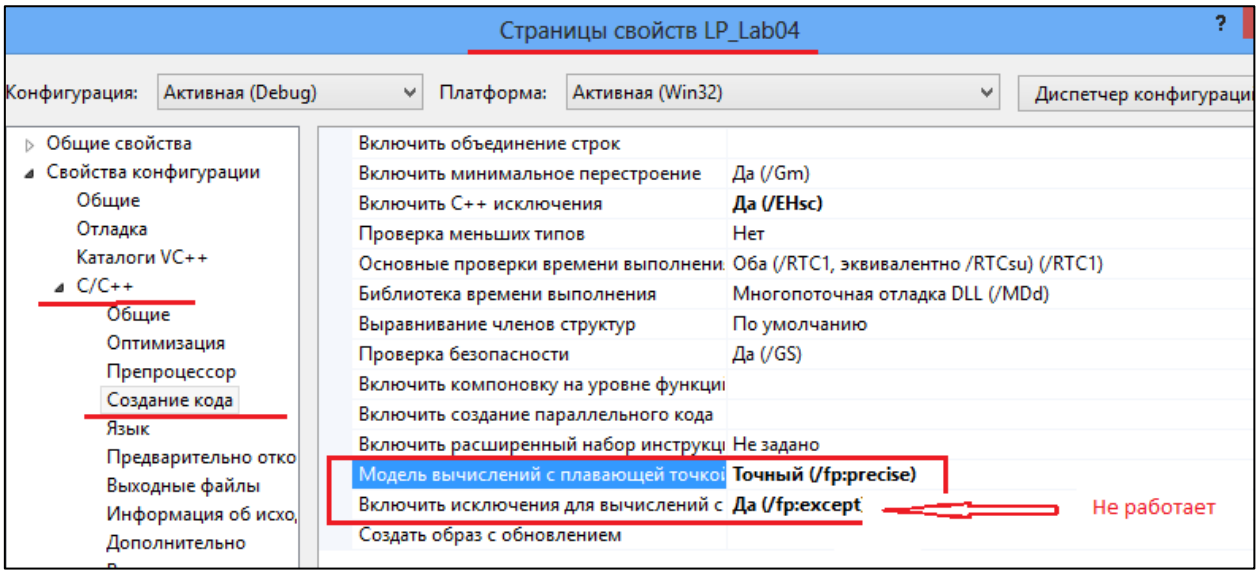



#include "stdafx.h"
#include <iostream>
// IEEE 754-1985 Standard for Binary Float-Point Arithmetic
int _tmain(int argc, _TCHAR* argv[])
{
int ld = sizeof(double);
int lf = sizeof(float);
float f1 = 23.625f;
float f2 = - 23.625f
double d1 = 1.234;
double d2 = -1.234;
double d3 = 1.234E5;
double d4 = 1.234E-5
Память 2
Адрес: 0x0033F860
0x0033F860 fc 80 93 af fc e0 e9 3e
0x0033F868 cc cc cc cc cc cc cc cc
0x0033F870 00 00 00 00 80 20 fe 40
0x0033F878 cc cc cc cc cc cc cc cc
0x0033F880 58 39 b4 c8 76 be f3 bf
0x0033F888 cc cc cc cc cc cc cc cc
0x0033F890 58 39 b4 c8 76 be f3 3f
0x0033F898 cc cc cc cc cc cc cc cc
0x0033F8A0 00 00 bd c1 cc cc cc cc
0x0033F8A8 cc cc cc cc 00 00 bd 41
0x0033F8B0 cc cc cc cc cc cc cc cc
0x0033F8B8 04 00 00 00 cc cc cc cc
0x0033F8C0 cc cc cc cc 08 00 00 00
0x0033F8C8 cc cc cc cc 1c f9 33 00
0x0033F8D0 09 4d c1 00 01 00 00 00
0x0033F8D8 40 9f 71 00 00 cf 71 00
0x0033F8E0 1a aa 06 41 00 00 00 00
Контрольные значения 1
Имя
Значение
ld
0x00000008
lf
0x00000004
&d2
0x0033f880 {-1.23
&d3
0x0033f870 {12340
&d4
0x0033f860 {1.234
&f1
0x0033f8ac {23.62
&f2
0x0033f8a0 {-23.6



Стандарт представления значений с плавающей точкой (IEEE 754) оставляет несколько зарезервированных значений, соответствующих **не-числам** (NaN, not-a-number). Стандартная библиотека Visual C++ печатает не-числа следующим образом:

Печатается	**Означает**
1.#INF	Положительная бесконечность
-1.#INF	Отрицательная бесконечность
1.#SNAN	Положительное сигнальное не-число *(signaling NaN)*
-1.#SNAN	Отрицательное сигнальное не-число *(signaling NaN)*
1.#QNAN	Положительное несигнальное не-число *(quiet NaN)*
-1.#QNAN	Отрицательное несигнальное не-число *(quiet NaN)*
1.#IND	Положительная неопределённость
-1.#IND	Отрицательная неопределённость




```
#include "stdafx.h"
#include <iostream>
// IEEE 754-1985 Standard for Binary Float

int _tmain(int argc, _TCHAR* argv[])
{

    int lf = sizeof(float);
    int ld = sizeof(double);
    int lld =  sizeof(long double);
```

Контрольные значения 1

Имя	Значение
lf	0x00000004
ld	0x00000008
lld	0x00000008