Лекция 02

БГТУ, ФИТ, ПОИТ, 2 семестр,

Конструирование программного обеспечения

Структура языка программирования

План лекции:

Структура языка программирования

- ✓ **алфавит языка**: набор разрешенных символов, кодировка символов исходного кода программ; символы времени трансляции, символы времени выполнения;
- ✓ **идентификаторы**: правила образования идентификаторов; зарезервированные идентификаторы;
- ✓ литералы;
- ✓ **ключевые слова**;
- ✓ **фундаментальные (встроенные) типы данных**:
  - предопределенные типы данных;
- ✓ пользовательские типы данных
  - типы, которые может создавать пользователь на основе фундаментальных типов (возможно описание их свойств и поведение);
  - массивы фундаментальных типов;
- ✓ **преобразование типов**: явное и неявное (автоматическое).
- ✓ **инициализация памяти:** присвоение значения в момент объявления переменной;
- ✓ **константное выражение:** выражение, которое должно быть вычислено на этапе компиляции;
- ✓ **область видимости переменных:** доступность переменных по их идентификатору в разных частях программы**;** пространства имен**;**
- ✓ **выражения**
- ✓ **инструкции языка:** инструкция — это некое элементарное действие, несколько идущих подряд инструкций образуют блок вычислений (последовательность инструкций);
  - присваивания;
  - инструкции объявления;
  - блок вычислений;
  - ветвление;
  - циклы;
  - инструкции перехода;
  - обработка исключений;
- ✓ **программные конструкции** (декомпозиция программного кода): процедуры, функции, методы, ...

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 программирования:

набор символов, разрешенных к использованию языком программирования. Основывается на одной из кодировок.

Совокупность символов, допускаемых в языке – алфавит языка.

Базовый набор символов исходного кода:

1) строчные и прописные буквы латинского и национального алфавитов
2) цифры
3) знаки операций
4) символы подчеркивания _ и пробела
5) ограничители и разделители
6) специальные символы

С помощью символов алфавита записываются **служебные слова**, которые составляют словарь языка.

Алфавит языка программирования служит для построения слов в языке программирования, которые называют лексемами. Примеры лексем:

Лексемы	***идентификаторы;*** ***ключевые (зарезервированные) слова;*** ***знаки операций;*** ***константы;*** ***разделители*** *(скобки, знаки операций, точка, запятая, пробельные символы и т.д.).*

Границы лексем определяются с помощью других лексем, таких, как ***разделители*** или ***знаки операций***.

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

   Набор символов времени трансляции:
   текст программы на языке программирования хранится в исходных файлах и основан на определенной кодировке символов.

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

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

   Стандарт C++: исходной код основывается на множестве символов ASCII:
   - ***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 С.

> Во время выполнения можно установить кодовую страницу языкового стандарта, используя вызов setlocale(LC_CTYPE, "rus")
>
> или
>
> воспользоваться следующими функциями, необходимо включить заголовочный файл <windows.h>:
>
> ```
> #include <windows.h>    // windows.h содержит прототипы функций
> SetConsoleOutputCP(1251);//установить кодовую таблицу, на поток ввода
> SetConsoleCP(1251);          //установить кодовую таблицу, на поток вывода
> ```
>
> 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.

Примеры правил составления идентификаторов в языках программирования:

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 в любом регистре 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.

Длина идентификатора по стандарту не ограничена.

Идентификатор создается при объявлении переменной, функции, типа и т. п.

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 зарезервированным;
- Кроме того: **is**xxxx, **mem**xxxx, **str**xxxx, **t**oxxxx, **wcs**xxxx, **Ецифра**xxxx, **LC_X**xxx, **SIGX**xxx, **SIG_X**xxxx.

9. Литерал:

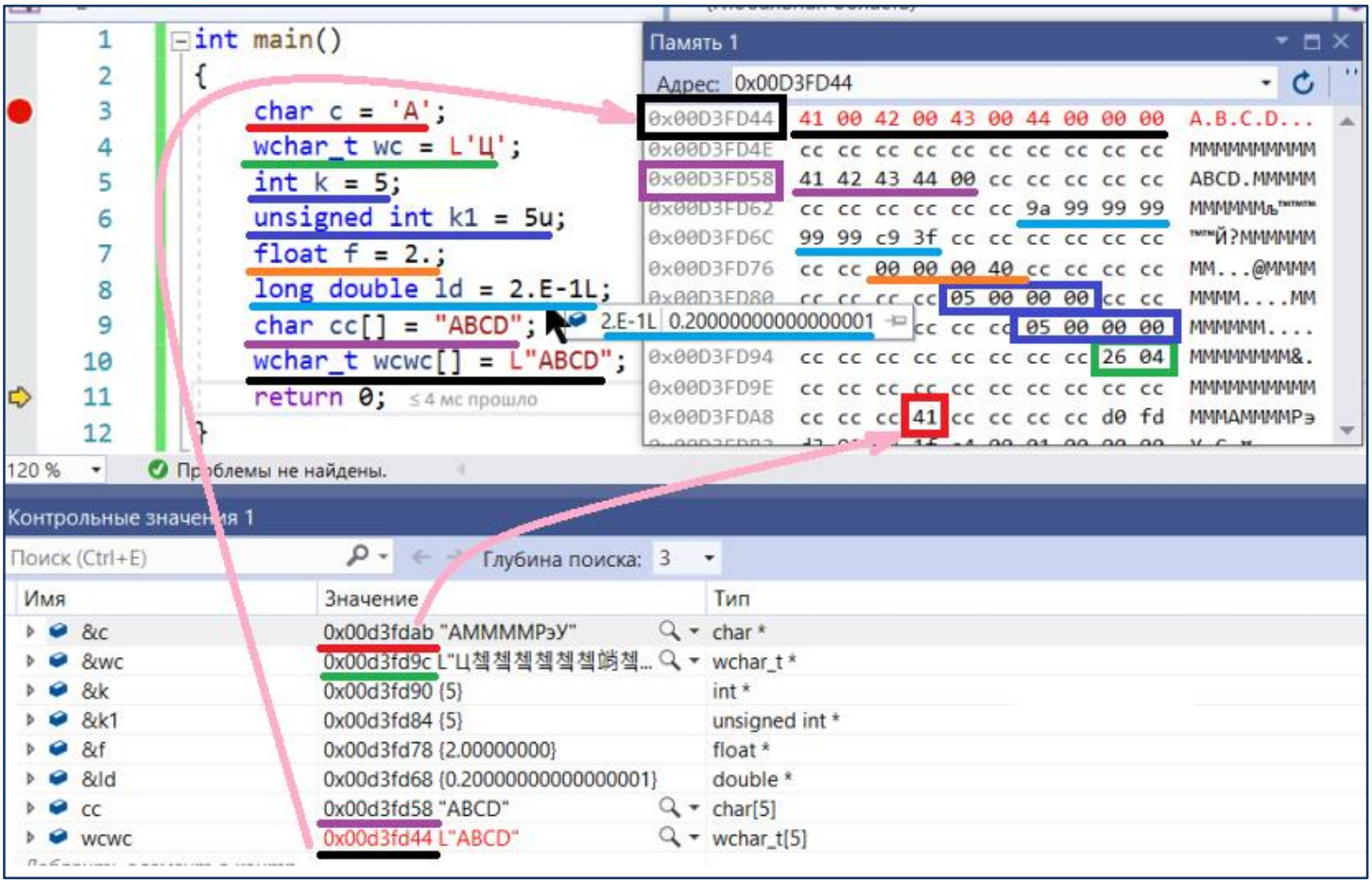
элемент программы, который непосредственно представляет значение.

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	?	



10. Ключевые слова:

последовательности символов алфавита языка, имеющие специальное назначение.

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

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, примитивные);
- 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.

Фундаментальные типы (или встроенные типы) задаются стандартом языка и встроены в компилятор.

Java 8

примитивные типы: 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

Фундаментальные типы 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 встроенных типов в Microsoft C++
(тип long имеет размер 4 байта даже в 64-разрядных операционных системах)**:**

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

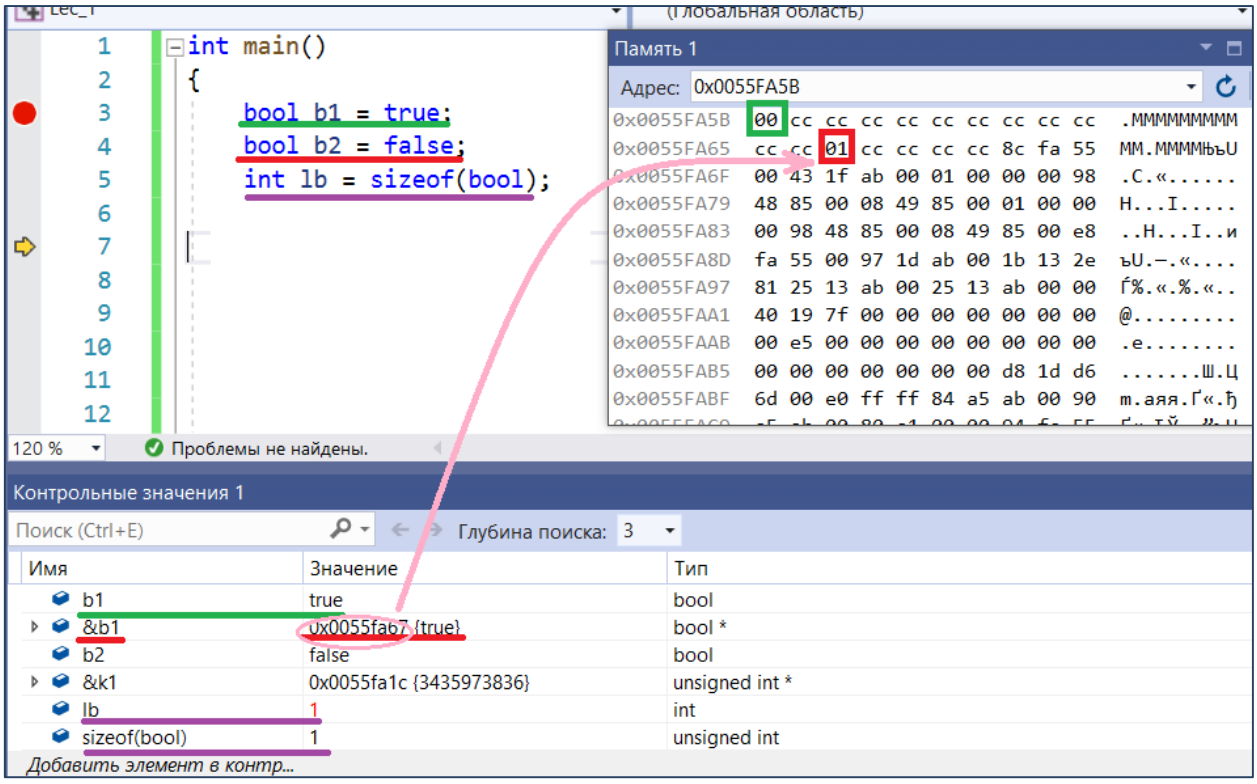
Диапазоны типов данных в компиляторах Microsoft C++:
https://docs.microsoft.com/ru-ru/cpp/cpp/data-type-ranges?view=msvc-160

Большинство встроенных типов имеют размеры, определенные реализацией.

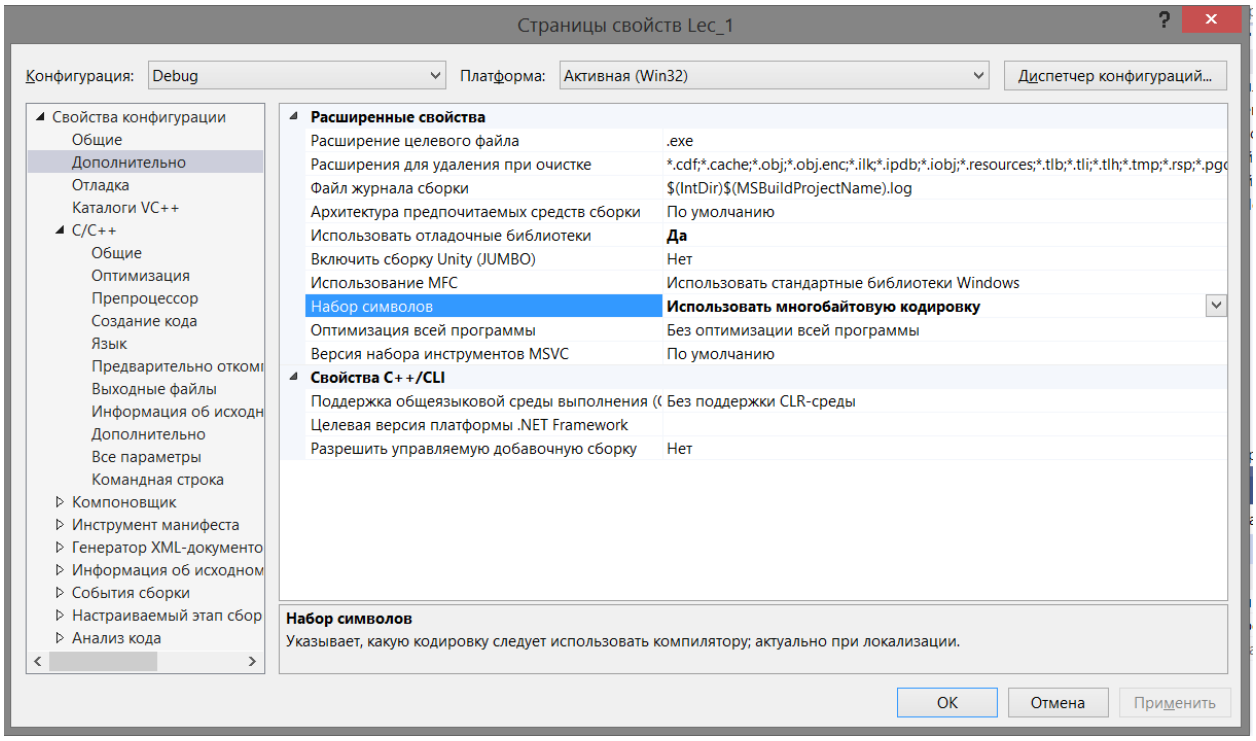
13. Фундаментальные типы C++. Примеры.

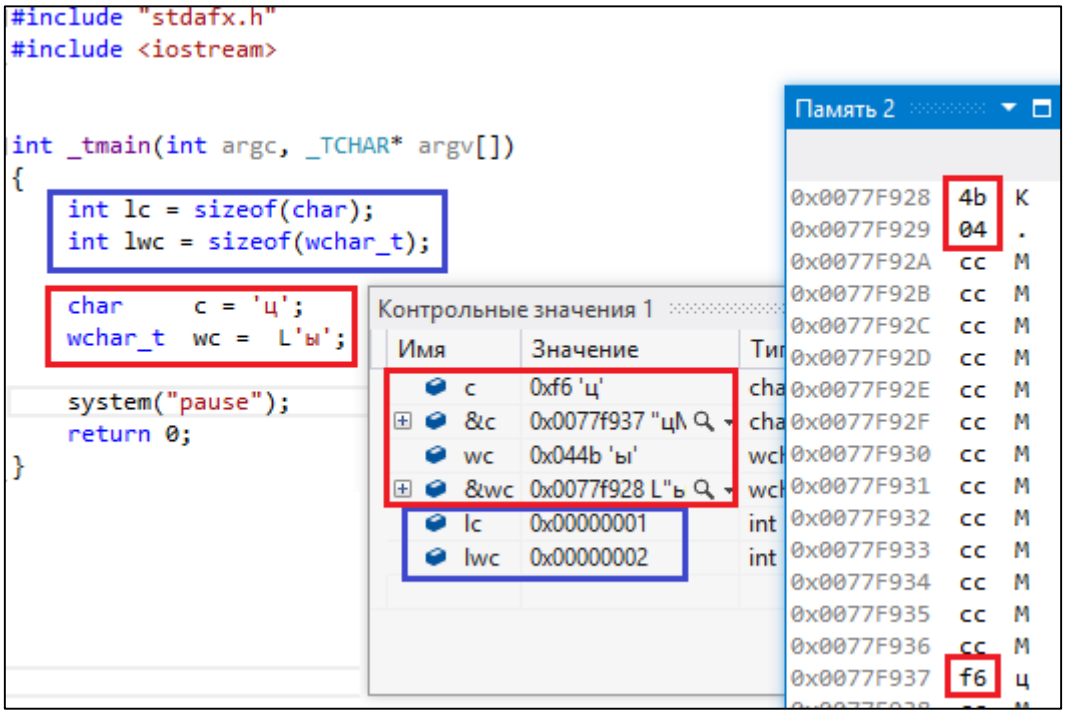
13.1 Тип bool

Размер типа bool зависит от конкретной реализации.



13.2 Встроенные типы char, wchar_t



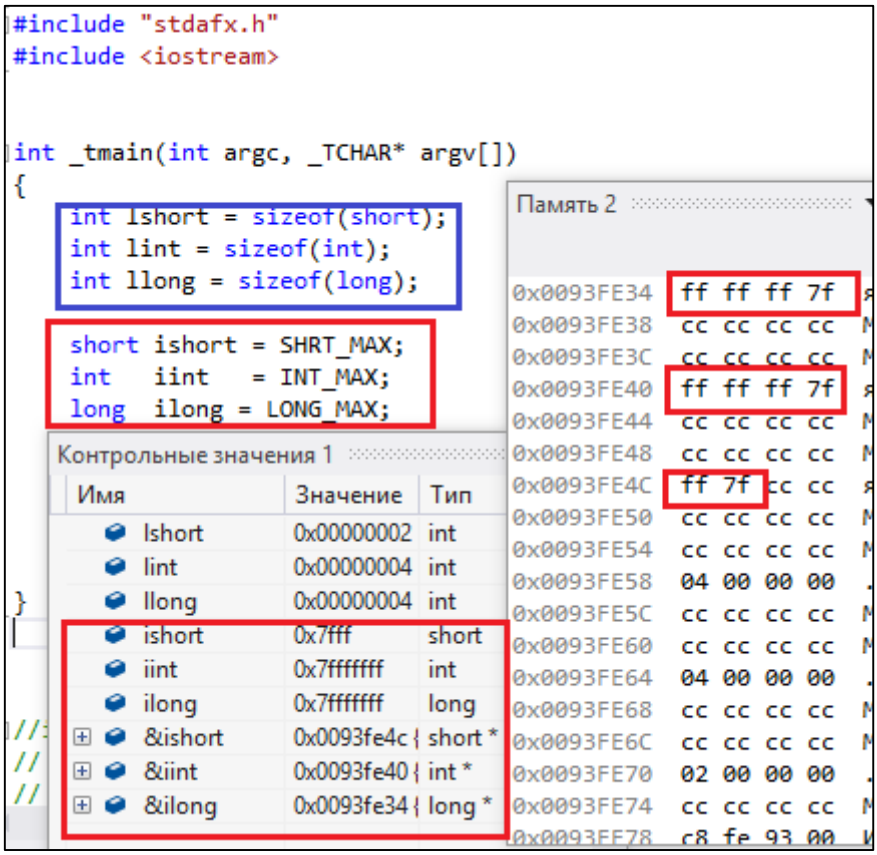


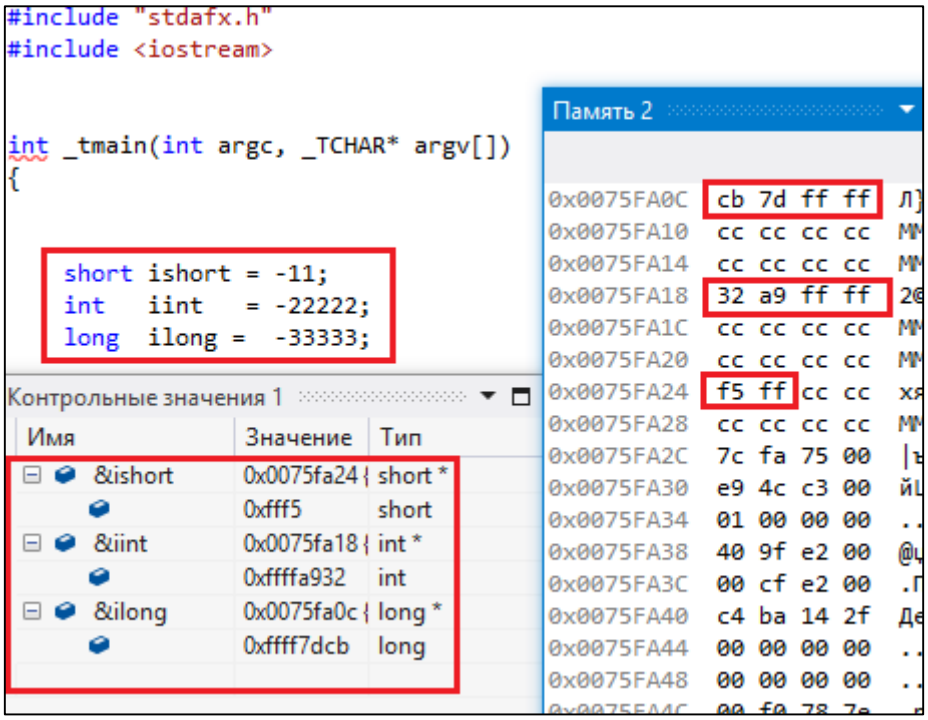
13.2.1 [unsigned] int, [unsigned] short, [unsigned] long

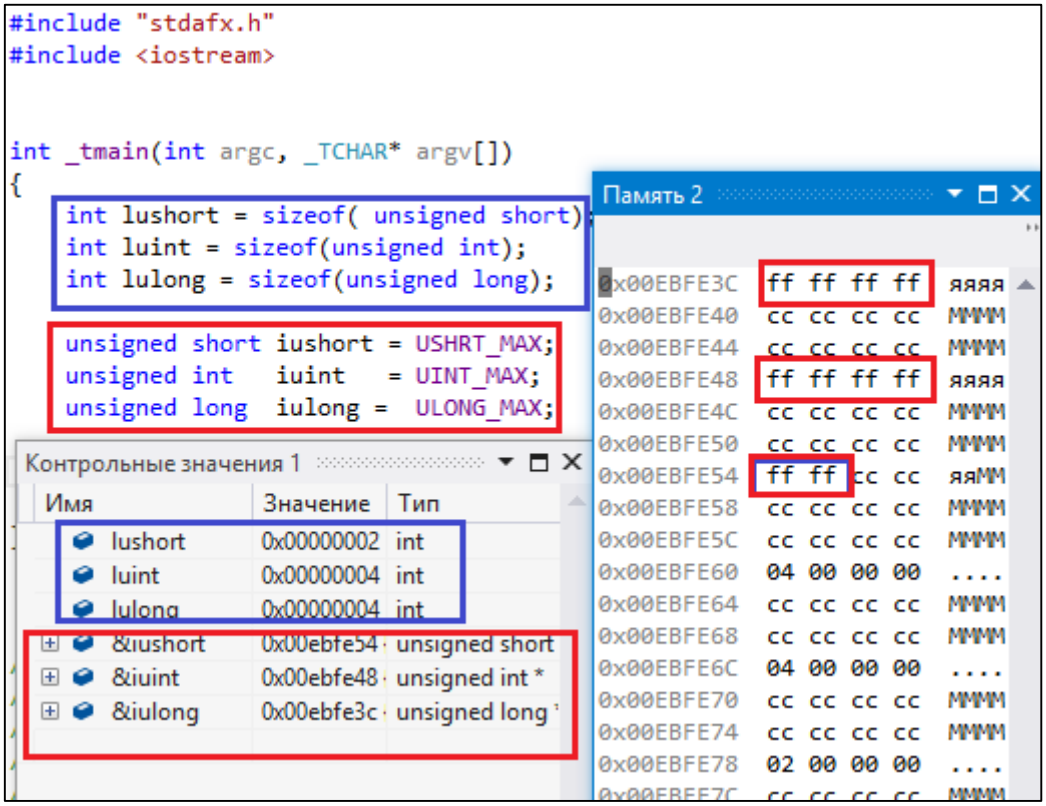
Внутреннее представление величины целого типа:

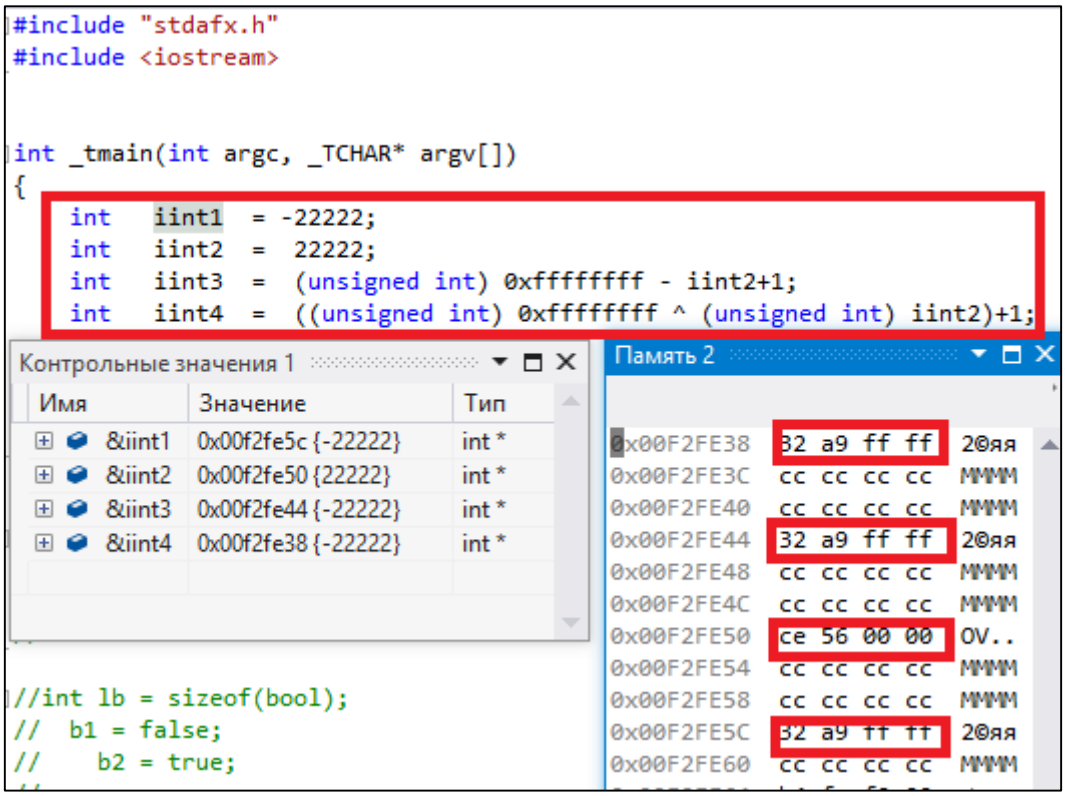
- целое число в двоичном коде.
- ***спецификатор*** **signed** – старший разряд (бит) числа интерпретируется как знаковый (**0** – положительное число, **1** – отрицательное).
- ***спецификатор*** **unsigned**: старший разряд (бит) рассматривается как значащий, позволяет представлять только положительные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

По умолчанию все целочисленные типы считаются знаковыми, то есть спецификатор **signed** можно опускать. Диапазон значений зависит от реализации.




#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)

13.3 Типы float, double

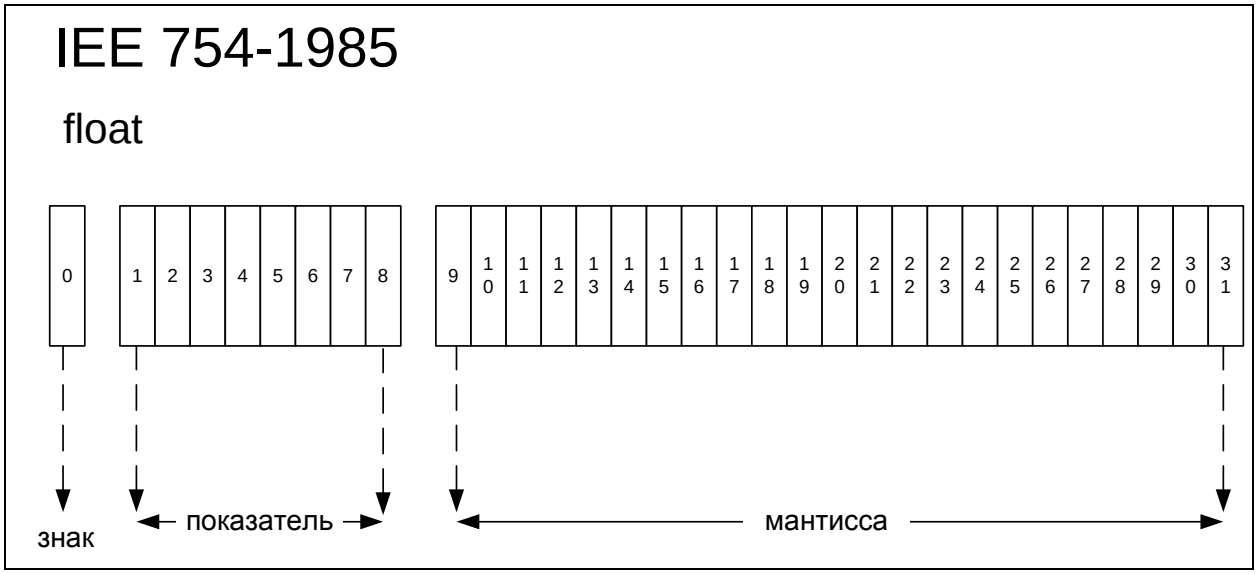
Стандарт языка C++ определяет три типа данных для хранения вещественных значений: ***float, double*** и ***long double***.

Стандарт IEEE 754 описывает формат представления чисел с плавающей точкой. Используется в программных (компиляторы разных языков программирования) и аппаратных (CPU и FPU) реализациях арифметических действий (математических операций).

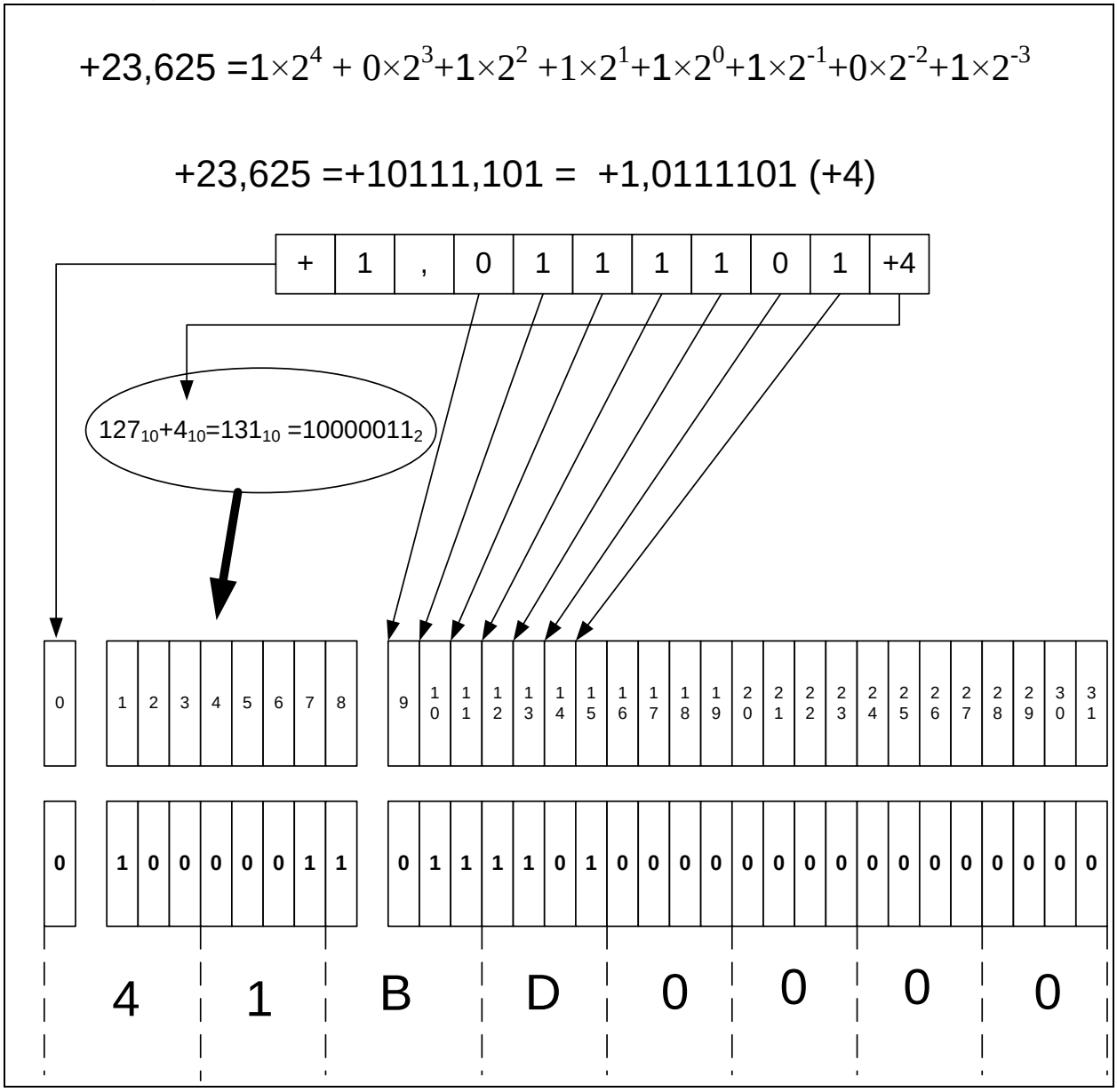
Стандарт описывает:

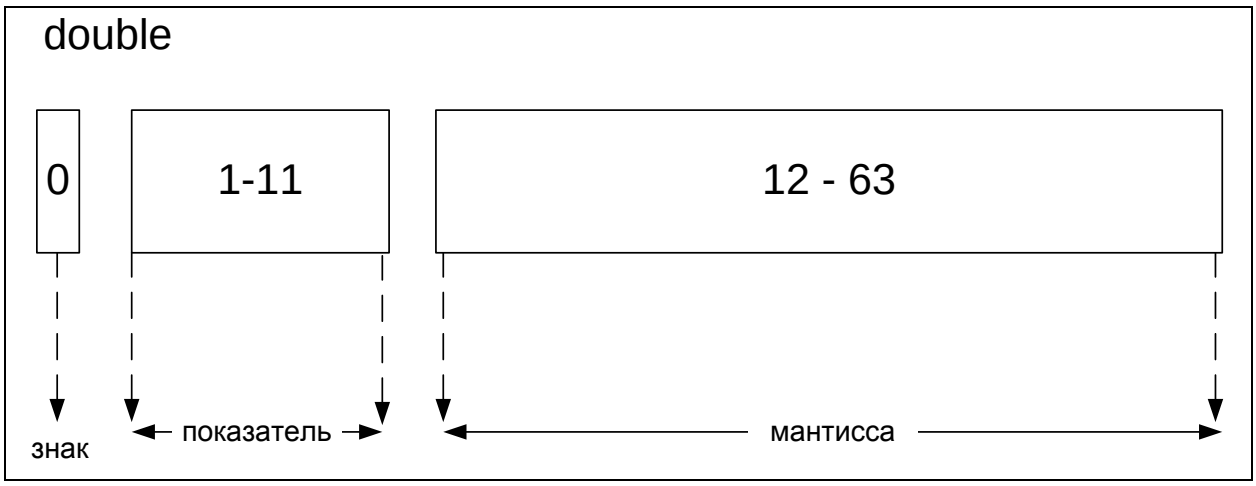
– формат чисел с плавающей точкой: мантисса, показатель (экспонента), знак числа;
– представление положительного и отрицательного нуля, положительной и отрицательной бесконечностей, а также нечисла (англ. Not-a-Number, NaN);
– методы, используемые для преобразования числа при выполнении математических операций;
– исключительные ситуации: деление на ноль, переполнение, потеря значимости, работа с денормализованными числами и другие;
– операции: арифметические и другие.

Стандарт 2008 года заменяет IEEE 754-1985. В новый стандарт включены двоичные форматы из предыдущего стандарта и три новых формата.



$23{,}625_{10} = 10111{,}101_2 = 1{,}0111101_2(+4)$
$23_{10} = 10111_2$
$0{,}625_{10} = 0{,}101$






```
#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;
```

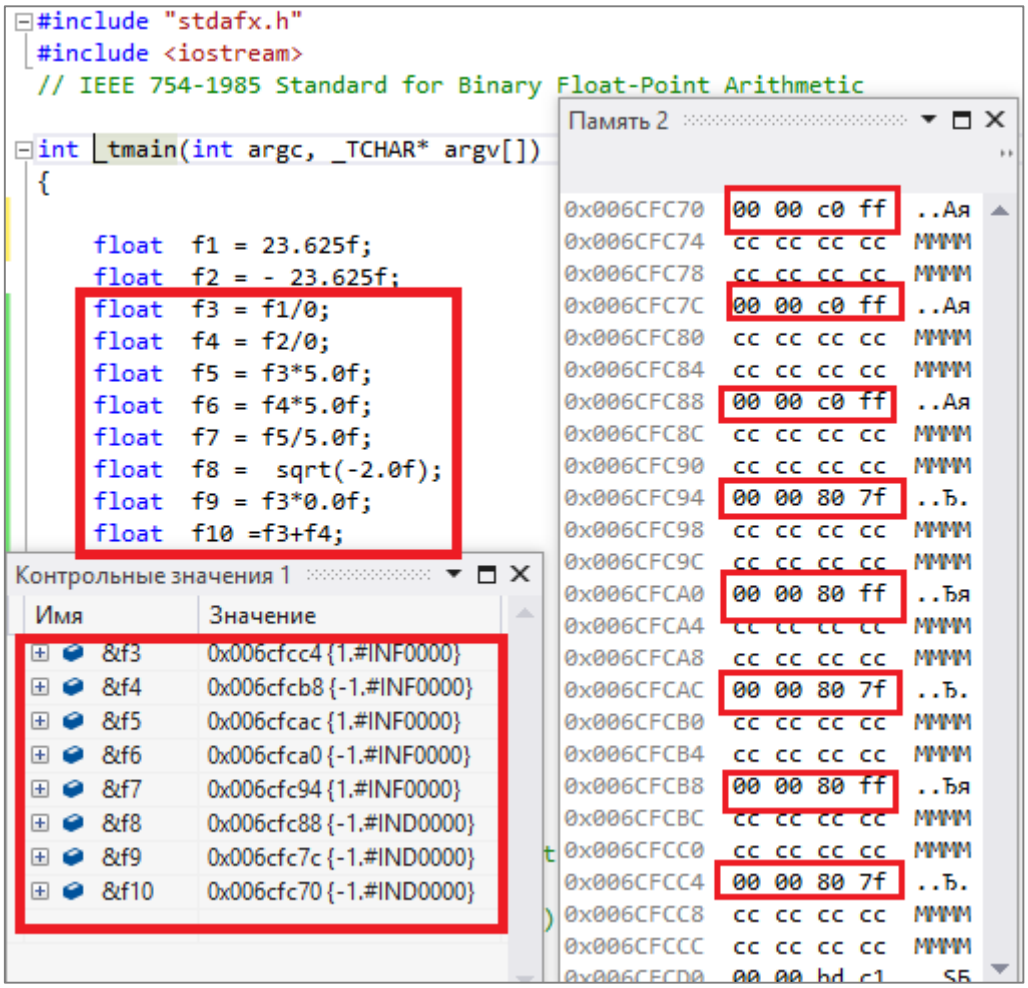
Контрольные значения 1

Имя	Значение
ld	0x00000008
lf	0x00000004
&d1	0x0033f890 {1.234
&d2	0x0033f880 {-1.23
&d3	0x0033f870 {12340
&d4	0x0033f860 {1.234
&f1	0x0033f8ac {23.62
&f2	0x0033f8a0 {-23.6

Память 2

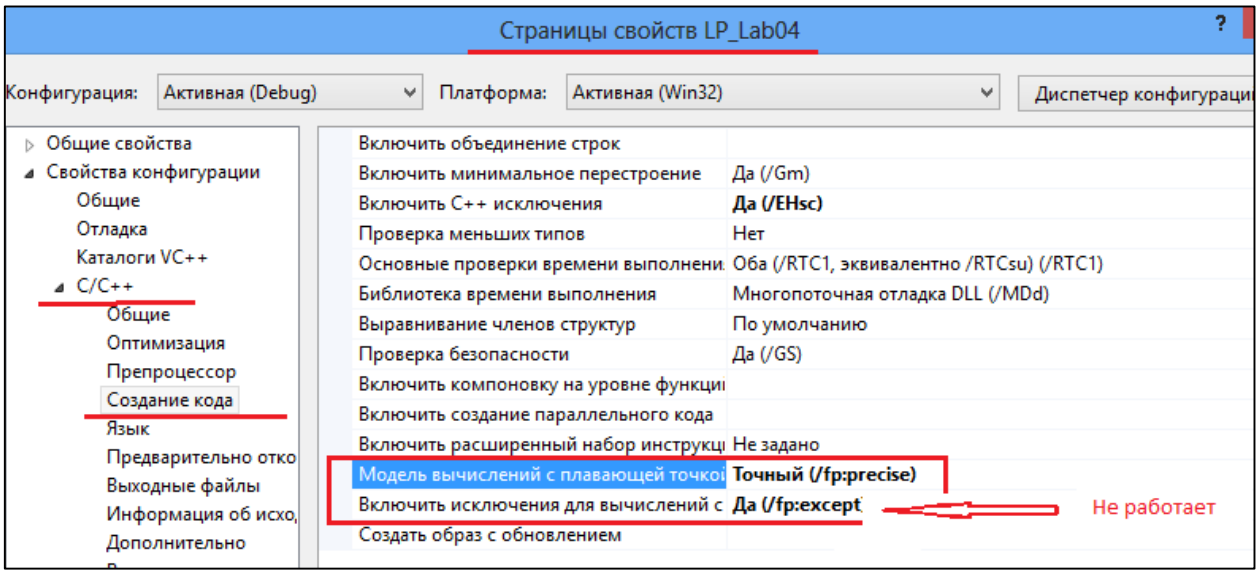
Адрес: 0x0033F860

```
0x0033F860  fc 80 93 af fc e0 e9 3e  ьЂ“Їьай>
0x0033F868  cc cc cc cc cc cc cc cc  ИИИИИИИИ
0x0033F870  00 00 00 00 80 20 fe 40  ....Ђ ю@
0x0033F878  cc cc cc cc cc cc cc cc  ИИИИИИИИ
0x0033F880  58 39 b4 c8 76 be f3 bf  X9ґИvsyї
0x0033F888  cc cc cc cc cc cc cc cc  ИИИИИИИИ
0x0033F890  58 39 b4 c8 76 be f3 3f  X9ґИvsy?
0x0033F898  cc cc cc cc cc cc cc cc  ИИИИИИИИ
0x0033F8A0  00 00 bd c1 cc cc cc cc  ..SБИИИИ
0x0033F8A8  cc cc cc cc 00 00 bd 41  ИИИИ..SA
0x0033F8B0  cc cc cc cc cc cc cc cc  ИИИИИИИИ
0x0033F8B8  04 00 00 00 cc cc cc cc  ....ИИИИ
0x0033F8C0  cc cc cc cc 08 00 00 00  ИИИИ....
0x0033F8C8  cc cc cc cc 1c f9 33 00  ИИИИ.щ3.
0x0033F8D0  09 4d c1 00 01 00 00 00  .МБ.....
0x0033F8D8  40 9f 71 00 00 cf 71 00  @џq..Пq.
0x0033F8E0  1a aa 06 41 00 00 00 00  .Є.A....
0x0033F8E8  00 00 00 00 00 b0 75 7f  .....°u.
```



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