Информация и данные

Информация – ругательное слово. Вместо него следует использовать слово данные – это такая информация, которую можно представить в определенном формате, передать и обработать программой

Информация как технический термин – физическая величина, может быть вычислена, используется как техническая характеристика.

Информация как философская (методологическая) категория – элемент описания общей картины окружающего мира

Немного философии

«Философия – попытка объяснения окружающего мира с точки зрения собственных слабостей и пристрастий.»
Романов Е.Л., вполне возможно, неосознанный плагиат.

«Всякая философия – самооправдание. Оригинальной была бы только философия, оправдывающая другого человека.» А.Камю, похоже, источник плагиата

«Не я плохой, а мир такой»
Перефразируя Камю

1. в программировании имеют место абстракции (идеальное) и их воплощение:
   - информация и носитель – «дырка от бублика», материальное или идеальное , бит и двоичный разряд
   - переменная = тип данных + память
   - код программы и экземпляр (runTime)
   - данные и мета-данные, система и мета-система (Java, СУБД)
2. Роль и место программы и программиста. Алгоритмически неразрешимые проблемы [cprog 3.8]

Подробнее. «Философия информации. Семантическая модель»
[cprog_cs: Метод.материалы и задания: Информатика]

Технические меры информации

Мера **неопределенности** – первая по времени возникновения, введена Шенноном в сугубо технической модели кодирования данных при их передаче. Исходит из того, что если имеется набор состояний системы-источника сообщений с известными вероятностями их появления, то сообщение о текущем состоянии системы содержит количество информации, как раз и определяемое формулой Шеннона $I = -\sum_{i=1}^{n} p_i \log_2 p_i$

Исходная постановка проблемы заключалась в определении *полосы пропускания канала связи и кодировании сигнала (т.е. формате передаваемых данных)*.
В дальнейшем было обнаружено сходство этой формулы с аналогичной в термодинамике, используемой для определения меры беспорядка, хаоса – **энтропии,** в связи с чем передача и получение информации стало трактоваться как снижение *информационной энтропии* системы.

Мера полезности, **прагматическая** – информация, содержащаяся в данных (сигнале) рассматривается по отношению к ценности (полезности) ее для принимающего субъекта

Мера «программируемости», **алгоритмическая** – наименее распространенная, основана на оценке сложности программы (машины Тьюринга), способной принимать, анализировать или воспроизводить информацию (данные).

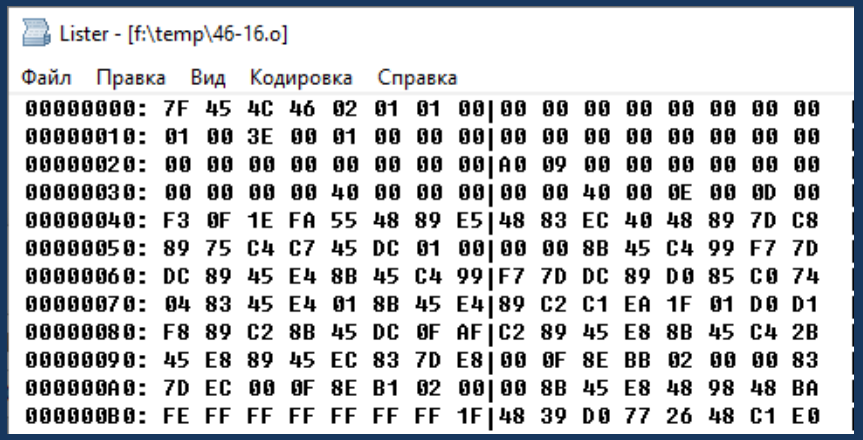
Меры информации

Мера многообразия, **синтаксическая** – наиболее распространенная в настоящее время, пресловутые мага- и гигабайты. Фактически та же самая мера неопределенности - энтропии, но для многообразия значений или состояний, о вероятностях появления которых ничего не известно, либо область применения меры настолько широка, что вероятности их появления считаются одинаковыми. Мера информации, выраженная в битах, для системы из N состояний, определяется как $I = \log_2 N$, а информационная емкость I двоичных разрядов – достаточна для кодирования $N = 2^I$ состояний

Мера осмысленности, **семантическая** – информация, содержащаяся в данных (сигнале) рассматривается по отношению к тому, как принимающий субъект может ее «понимать», технически ближе – интерпретировать, изменять свою структуру (в простейшем случае, запоминать)

Биты, байты, машинные слова

Физическая модель данных – представление любых хранимых и передаваемых данных в виде наборов двоичных битов без какой-либо смысловой (семантической) нагрузки. Соответствует *синтаксической мере информации.*



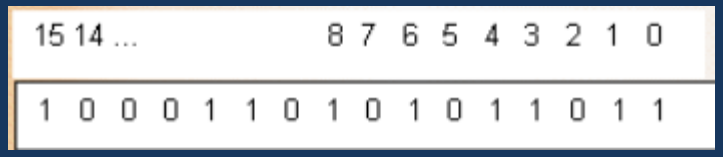
Бит – количество информации о равновероятном событии «да/нет».
В *синтаксической мере* один бит – это данные, передаваемые значением 1 или 0 о равновероятном событии «да/нет».
Двоичный разряд – элемент архитектуры, хранящий один бит данных

Двоичный разряд - стакан, бит – содержимое стакана.

В теории информации *(мера неопределенности)* – учитывается вероятность появления события, что влияет на количество информации, связанное с его появлением.

Биты, байты, машинные слова

Машинное слово – упорядоченное множество двоичных разрядов. Количество разрядов в машинном слове называется его **размерностью.** Разряды нумеруются справа налево, начиная с 0. Самый правый разряд называется младшим, самый левый – старшим.



Байт – машинное слово минимальной размерности, 8 двоичных разрядов.
Размерности всех других машинных слов кратны байту.
Стандартное машинное слово - машинное слово, размерность которого совпадает с разрядностью процессора.
Укороченное машинное слово - **short**
Двойное машинное слово – **long**

Замечание: разрядности бывают разные
- *разрядность процессора*
- *разрядность ОС* – стандартная размерность представления данных в ОС (x86-32, x64-64)
- *разрядность транслятора* – разрядность данных, с которой работает код
- *разрядность платформы* – Java – 32 (платформенно-независимая)

sizeof(int) – операция = разрядность в байтах типа данных или переменной для данной версии транслятора

Биты, байты, машинные слова

Информационная емкость МС (синтаксическая мера) – количество вариантов представления значений в МС, W=2^N

$2^{10} = 1024 \approx 1000 = 10^3$

«Чайник – это тот, кто думает, что в килобайте 1000 байтов, а программист – это тот, кто считает, что в килограмме 1024 грамма». **Анекдот (Здесь на самом деле больше смысла, чем юмора)**.

Информационная емкость трех десятичных цифр (разрядов) примерно соответствует аналогичной емкости десяти двоичных разрядов:

- 10^3 – кило ≈ 10 разрядов
- 10^6 – мега ≈ 20 разрядов
- 10^9 – гига ≈ 30 разрядов
- 10^12 – тера ≈ 40 разрядов
- 10^15 – пета ≈ 50 разрядов

Пример: 32 разрядное МС = 2^32 = 2^(2+30) ≈ 4*10^9 = 4 млрд

Помним степени 2:

2^8 = 256

2^16 = 65536

Системы счисления

«Цифры – это такие маленькие числа, которые используются для представления больших чисел». Определение

Позиционная система счисления (СС) – значение числа определяется в зависимости расположения цифр в его позициях:

• W= ∑Rk * N^k - k=0…M – позиция числа, начиная с 0, справа налево (по-арабски!!!!)

N – основание системы счисления, Ri = 0…N-1 – цифра в i-ой позиции

Популярные СС: 10-ая, 2-ая, 8-ая, 16-ая (0…9A…F), 12-ая (one…ten,eleven,twelfe), дюжина.

Преобразование целого в десятичную систему счисления. Для перевода целого числа, представленного в системе счисления с основанием **P**, нужно воспользоваться формулой определения значения числа в этой системе счисления, выполнив соответствующие действия над цифрами (в родной десятичной системе) - $W= \Sigma R_i p^i$.

$2C5_{16} = 2 \cdot 16^2 + 12(C) \cdot 16^1 + 5 = 512 + 192 + 5 = 709_{10}$

$4375_8 = 4 \cdot 8^3 + 3 \cdot 8^2 + 7 \cdot 8 + 5 = 2048 + 192 + 56 + 5 = 2301_{10}$

Преобразования дробной части числа в десятичную систему счисления. Для дробной части числа нужно использовать ту же самую формулу с учетом отрицательных степеней основания для разрядов дробной части - $W= \Sigma R_i p^{-i}$.

$0.3F5_{16} = 3/16 + 15(F)/16^2 + 5/16^3 = 0.18750 + 0.05859 + 0.00122 = 0.24731_{10}$

$0.524_8 = 5/8 + 2/8^2 + 4/8^3 = 0.6250 + 0.0312 + 0.0078 = 0.6640_{10}$

Системы счисления

Преобразование целого из десятичной системы счисления в систему с основанием P. Идея алгоритма заключается в том, что остаток от деления исходного числа на **P** дает нам младшую цифру числа в этой системе счисления. Последовательность остатков от деления исходного числа на основание системы счисления образует цифры числа, но в обратном порядке, начиная с младшей.

2301 / 8 = 287 (5)
287 / 8 = 35 (7)
35 / 8 = 4 (3)
4 / 8 = 0 (4) = 4375_8

709 / 16 = 44 (5)
44 / 16 = 2 (12=C)
2 / 16 = 0 (2) = $2C5_{16}$

Преобразование дробной части из десятичной системы счисления в систему с основанием P. При умножении дробной части на основание системы счисления **P** очередная цифра дробной части «выдвигается» в целую часть числа. Если ее отбросить и повторить процесс, то получим последовательность цифр дробной части (в прямом порядке).

0.24731 * 16 = 3.95696 (3)
0.95696 * 16 = 15.31136 (15(F))
0.31136 * 16 = 4.98176 (4)
0.98176 * 16 = 15.70816 (15(F)) = $0.3F4F_{16} \approx 0.3F5_{16}$

0.6640 * 8 = 5.312 (5)
0.312 * 8 = 2.496 (2)
0.496 * 8 = 3.968 (3)
0.968 * 8 = 7.744 (7) = $0.5237_8 \approx 0.524_8$

Обратите внимание, что при преобразовании дробной части числа в десятичную систему, а затем обратно в

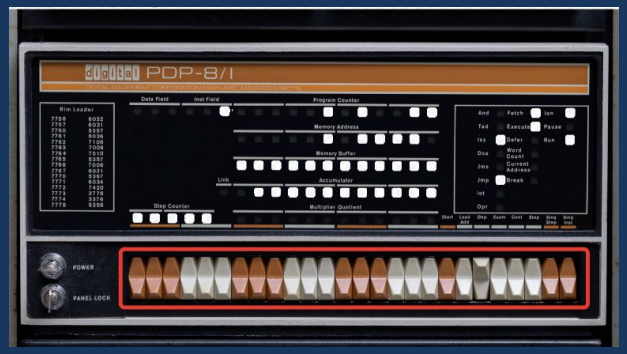
Эквиваленты двоичной СС

Используются для представления значения МС в их «первородном» виде (набор двоичных разрядов)

Основание кратно степени 2:

- восьмеричная – одна цифра 0…7 соответствует 3 разрядам (триада)
- шестнадцатеричная - одна цифра 0…7A…F соответствует 4 разрядам (тетрада)

	8421		8421		8421		8421
0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111



0xC0FFEE = 12 648 430

Исторические реалии:

- 8СС – 70-80 г., 16 разрядные МС 0000000…177777
- Строковые константы Си – коды символов в 8СС “\60\12\15”
- 16СС – начиная с микропроцессоров, 1 байт=2 цифры в 16СС 00..FF

Правила Си:

- 8-ая константа начинается с 0 - 0177 – константа в 8СС
- 16-ая константа начинается с 0x – 0x1FF

Грабли: 177 и 0177 – две большие разницы

Машинные слова в памяти и потоках

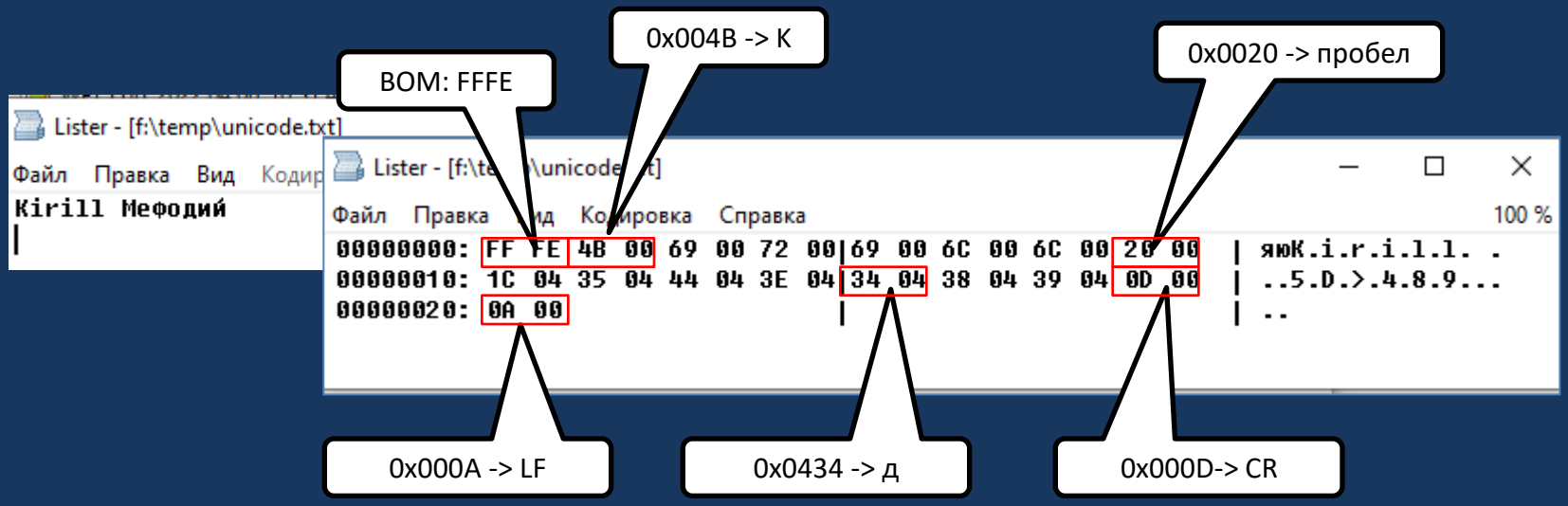
Направление просмотра (чтения):
- европейское – слева направо
- арабское – справа налево (ТОРА, КОРАН)

Последовательность передачи разрядов в последовательном потоке: начиная с младшего/начиная со старшего
Последовательность хранения в памяти и передачи байтов МС:
- младшим байтом вперед – **little-endian** – «интеловский», архитектура x86, форматы большинства мультимедиа-файлов [cprog 9.4]
- старшим байтом вперед – **big-endian –** потоки данных Java, формат протоколов TCP/IP, архитектуры IBM 360/370/390, SPARC, Motorola 68000

Термины big-endian и little-endian первоначально не имели отношения к информатике. В сатирическом произведении Джонатана Свифта «Путешествия Гулливера» описываются вымышленные государства Лилипутия и Блефуску, в течение многих лет ведущие между собой войны из-за разногласия по поводу того, с какого конца следует разбивать варёные яйца. Тех, кто считает, что их нужно разбивать с тупого конца, в произведении называют Big-endians («тупоконечники»).

Машинные слова в памяти и потоках

Дамп (dump) текстового файла: Unicode UTF-16 **LE** с BOM



Форматы представления данных

Машинное слово может содержать значение в некоторой форме представления в соответствии с принятым форматом: стандарт, система команд процессора (операнды), транслятор, программа при работе с данными в этом формате «напрямую», эмуляция – программная реализация формата представления при отсутствии его в «железе».

Форматы данных (архитектурные, базовые типы данных):

- целое без знака (целый тип с модификатором unsigned)
- целое со знаком – прямой код (---)
- целое со знаком – дополнительный код (архитектурный тип целого int)
- вещественное число с фиксированной точкой (запятой) – (fixed point) (---)
- вещественное число с плавающей точкой (запятой) – (floating point) (float,double)
- символ, строка, текст (char)
- логический (boolean) – представлен через int

Представление целого

Целое без знака (архитектурный тип)

Формат МС – все разряды значащие. Значение целого:

• сумма весов двоичных разрядов МС со значением 1

• по определению значения числа в позиционной СС - $W= \sum Rk * 2^k$

Диапазон: 0…2^N - 1

Представление в МС: 0…0xFFFFFFFF

Целое со знаком – прямой код (архитектурой и Си не поддерживается)

Формат МС – разряды 0…N-2-значащие. Старший разряд – знак (0 – “+”, 1-”-”)

Значение целого:

• абсолютное значение - по определению значения числа в позиционной СС

$W= \sum Rk * 2^k$ k=0..N-2

Диапазон: -2^(N-1) – 1…0…2^(N-1) - 1

Представление в МС:

0…0x00000000

1…0x00000001

…

2^(N-1) - 1 …0x7FFFFFFF

-1…0x80000001

-2…0x80000001

…

-2^(N-1) - 1 …0xFFFFFFFF

0x80000000 – «минус 0», неоднозначное значение

Представление целого

Целое без знака - дополнительный код (архитектурный тип)

- отображение диапазона отрицательных чисел на старшую половину диапазона целых без знака
- «фокус» работает во всех СС [cprog 1.3]

```
#include <stdio.h>
int main() {
    int a = 125, b;
    b = ~a+1;
    a = ~a;
    a++;
    printf("a=%d  %x\nb=%d %x\n", a, a, b, b);
    }
```

```
Консоль отладки Micr
a=-125  ffffff83
b=-125 ffffff83
```

Изменение знака целого в доп.коде в 2СС:

- инвертировать все разряды, включая знаковый
- добавить 1

Диапазон:

-2^(N-1) – 1…0…2^(N-1) - 1

Представление в МС:

0…0x00000000

1…0x00000001

…

2^(N-1) - 1 …0x7FFFFFFF

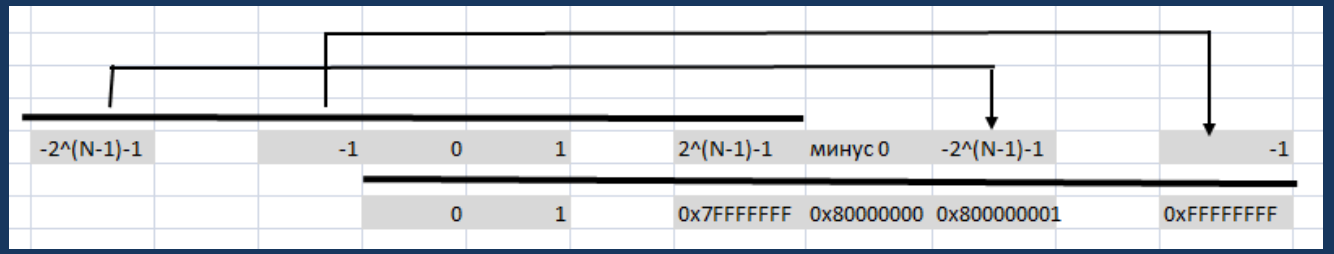
-1…0xFFFFFFFF

-2…0xFFFFFFFE

…

-2^(N-1) - 1 …0x80000001

0x80000000 – «минус 0»,



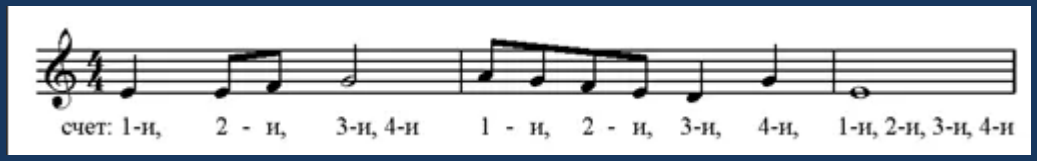
Дополнительный код - беззнаковая форма представления чисел со знаком.

Представление вещественного

Вещественное число с фиксированной точкой (архитектурой и Си не поддерживается)

- старший разряд знаковый
- последующие разряды слева направо имеют веса ½ , ¼ …

Диапазон: -1 < V < 1



Пример:

[0]101101 = 0.5 + 0.125 + 0.0625+0.015625 = 0.703

Вещественное число с плавающей точкой (архитектурный тип, float,double)

Вещественное число. Вещественное число (число в формате с плавающей запятой или с плавающей точкой) при его внешнем представлении может состоять из двух компонент: мантиссы – и порядка: **bbbb.bbbb** · 10^{p}. Одно и то же значение может быть представлено в разных видах, представление называется нормализованным, если мантисса не имеет целой части и все ее цифры в дробной части – значащие, например, $35.76 = 0.3576 \cdot 10^{2}$.

Представление вещественного

Во внутренним представлении дело обстоит аналогичным образом, но только с учетом двоичного представления всех данных:

- внутренний порядок основывается на степени 2, а не на степени 10, т.е. число выглядит как $\mathbf{1.b_1b_2b_3b_4…b_{n-1} \cdot 2^p}$. Порядок **p** является обычным целым числом со знаком
- мантисса представляет собой дробное нормализованное двоичное число. Разряды машинного слова, которые его содержат, нумеруются слева направо (от старшего к младшему). Значение мантиссы в таком представлении $\mathbf{W= \Sigma b_i\ 2^{-i}}$. Точка находится перед старшим разрядом, а сами разряды имею веса 1/2 , 1/4, 1/8, 1/16 и т д. слева направо
- нормализованная мантисса имеет в двоичном формате вид 1.XXXXXX

Реальный формат вещественного числа типа float (double) имеет несущественные отличия:

- вместо порядка в формате 8(11)-разрядного беззнакового целого хранится значение **p+1023** (double) или **p+127** (float) - всегда положительное
- в нормализованной мантиссе отбрасывается целая 1.
- знак мантиссы представлен отдельным разрядом.

Стандарт IEEE 754-2008

Представление вещественного

Программно структура формата float в виде МС видна.
Взламываем с помощью адресной арифметики [cprog 9.2]

```
int main() {
    float a = 1.25;
    int b;
    b = *(int*)&a; // float 4 байта = int 4 байта
    printf("b=%x\n", b);
    a = 2.5;
    b = *(int*)&a;
    printf("b=%x\n", b);
    a = 0.625;
    b = *(int*)&a;
    printf("b=%x\n", b);
}
```

```
Консоль отладки Mi
b=3fa00000
b=40200000
b=3f200000
```

Число одинарной точности (*Binary32, Single precision, float*) [править]

Число́ оди́нарной то́чности — компьютерный формат представления чисел, занимающий в памяти одно машинное слово (в случае 32-битного компьютера — 32 бита или 4 байта). Используется для работы с вещественными числами везде, где не нужна очень высокая точность.

Порядок записан со сдвигом −127.

https://habr.com/ru/post/112953/

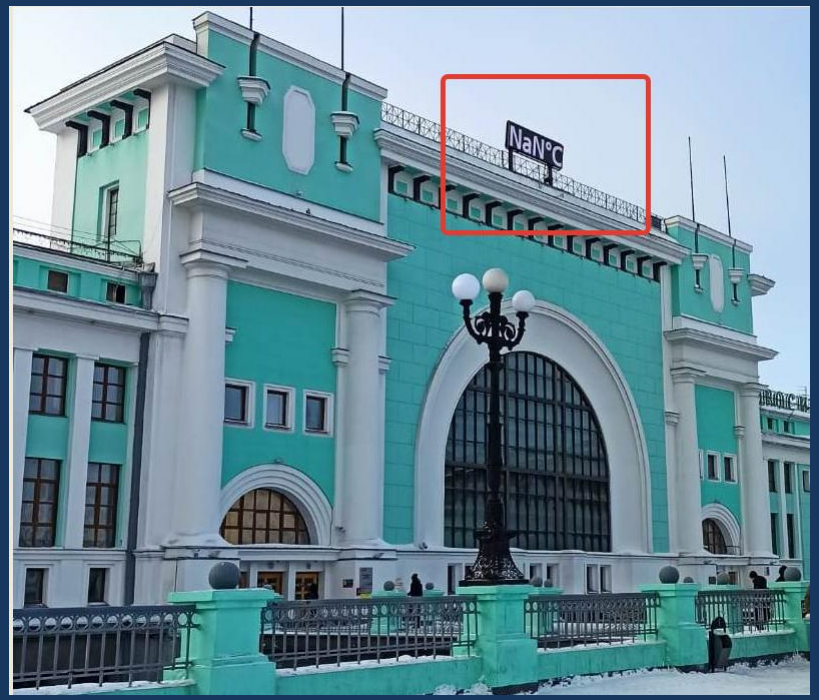
3 F A
0011 1111 1010 …
0[011 1111 1][010 …
Знак =0(+) Порядок-127= 127-127=0
Нормализованная мантисса со старшей 1 = 1.010…..= 1 + 0.25 = 1.25

4 0 2
0100 0000 0010 …
0[100 0000 0][010 …
Знак =0(+) Порядок-127= 128-127=1
Нормализованная мантисса со старшей 1 = 1.010…..= 1 + 0.25 = 1.25
1.25 * 2^1 = 2.5

Представление вещественного

Из формата вещественного числа следует, что *вещественное число по своей природе не всегда представлено точным значением*. Оно имеет ограниченное количество точных значащих цифр, но при этом не всякое точное десятичное число имеет соответствующее точное внутреннее представление (условно говоря, значение 0.1 окажется представленным ближайшими к нему 0.099999997… или 0.100000001…), имеющими точное внутреннее представление.

Ошибка: сумма вещественных «проскакивает» через точное значение 1 (Java)



-30 – криворукие программисты???

```
double dd=1/33.,vv=0;
System.out.println("dd="+dd);
//-------- Сумма пролетает через 1
for (vv=0;vv!=1;vv+=dd){
    if (vv>1) break;
    }
System.out.println(vv);
int i=0;
//--------- 33 раза по 1/33 меньше 1
for (vv=0,i=0;i<33;vv+=dd,i++);
System.out.println(vv);
System.out.println(dd*33.);
}
```

```
dd=0.030303030303030304
1.0303030303030296
0.9999999999999993
1.0
```

Тип данных, переменная, значение

Представление данных в соответствии с *семантической (смысловой)* мерой информации

Тип данных (ТД) - описание данных определенного вида, для которых известен их способ представления в памяти (формат), следующие из него размерность и диапазон значений, а также определен набор операций.

Тип данных – это «идеи» о представлении переменных определенного вида, и они должны быть известны транслятору (для генерации необходимого кода) и программисту.

Тип данных = формат + размерность, диапазон значений + операции

Базовые типы данных (БТД) заложены в транслятор «от рождения». Ориентированы на область применения языка. В Си БТД совпадают с основными форматами представления данных в компьютерных архитектурах**,** т.е. операции над ними «один в один» транслируются в соответствующие машинные команды. Естественно, что БТД обозначаются в любом языке **ключевыми словами.**

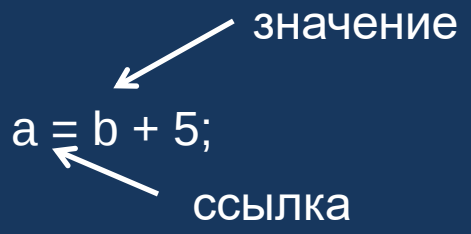
Производные типы данных (ПТД) конструируются в программе из уже известных в программе базовых и производных типов (иерархия типов, некоторые из них могут обозначаться дополнительными именами, которые можно использовать синтаксически как базовые. ПТД: массивы, в Си++ - классы.

Тип данных, переменная, значение

Переменная – область памяти, в которой содержатся данные определенного типа. Имя переменной напрямую ассоциируется с ее адресом, содержимое памяти – со значением переменной.

Переменная = память (имя, адрес,ссылка) + ТД + значение

Дуализм имени переменной:
- значение переменной
- местоположение переменной (адрес, ссылка)



Си является языком *строго типизированным*. Использование предварительно не определенных программных объектов, в том числе переменных, не допускается (т.е. невозможна привязка типа автоматически, по умолчанию)
Си является языком со *статическим определением* типа. Тип переменной определяется при трансляции и не может быть изменен при исполнении

Философия: тип данных – абстрактная категория, переменная – реализация (экземпляр) типа данных в однородной физической среде (памяти)

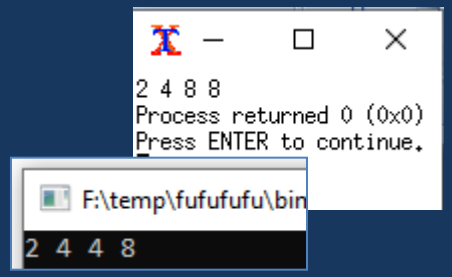
Синтаксис определения переменных

Определение – это текстовое описание объекта программы (переменной, функции, класса), по которому транслятор создает его внутреннее (двоичное) представление (размещает в памяти). Определение объекта должно производиться в программе только один раз, а соответствующий ему внутренний программный код размещается в создаваемом при трансляции объектом модуле.

Объявление – сообщение транслятору об имени и свойствах объекта программы (переменной, функции), который в данной точке недоступен (неизвестен транслятору), но который используется программой.

Базовые типы данных:

- char – целое со знаком, байт, символ в байтной кодировке
- int – целое со знаком, стандартное МС, 4 байта
- short – короткое целое со знаком, 2 байта
- long – целое (длинное) со знаком, 4 (8) байтов
- long long – длинное целое со знаком, (8) байтов
- float/double – короткое и обычное вещественное (4/8)
- void – пустой тип, заглушка



Платформенная зависимость компилятора: Windows long – 4 байта, Linux (WSL) – 8

```c
#include <stdio.h>

int main(){
    printf("%d %d %d %d", sizeof(short), sizeof(int), sizeof(long), sizeof (long long));
    return 0;
    }
```

Базовые ТД и константы

```
int main() {
    int a = 1256;                    // Целая константа
    short b = 32000;
    long long c = 1000000000000L;    // "Длинная" константа
    unsigned uu = 333333;
    char ss = 125;
    printf("a=%d b=%d c=%lld uu=%u ss=%d\n", a, b, c, uu, ss);
    std::cout << "a=" << a << " b=" << b << " c=" << c << " uu=" << uu << " ss=" << (int)ss << "\n";
    a = 01001;                       // Восьмеричная константа
    b = 0x101;                       // 16-ричная константа
    ss = '*';                        // Символьная константа (char)
    printf("a=%d b=%d ss=%c\n", a, b, ss);
    std::cout << "a=" << a << " b=" << b << " ss=" << ss << "\n";
    // ПОЧЕМУ ????????
    a = (int)3000000000;         // Переполнение - диапазон чисел <0
    b = (short)65538;            // Переполнение - потеря разрядов
    uu = -1;                     // Преобразование signed-unsigned с сохранением МС
    ss = (char)257;              // Переполнение - потеря разрядов
    printf("a=%d b=%d c=%lld uu=%u ss=%d\n", a, b, c, uu, ss);
}
```

```
Консоль отладки Microsoft Visual Studio
a=1256 b=32000 c=1000000000000 uu=333333 ss=125
a=1256 b=32000 c=1000000000000 uu=333333 ss=125
a=513 b=257 ss=*
a=513 b=257 ss=*
a=-1294967296 b=2 c=1000000000000 uu=4294967295 ss=1
```

- модификаторы констант L,U, префиксы 0,0x
- форматированный вывод stdio [cprog 4.5] – контроль соответствия типов в printf
- форматированный вывод iostream – переопределение операции << в C++
- диапазоны значений для целых ТД, переполнение, потеря разрядов
- преобразования при смене ТД
- char – байт, целое, символ (вывод)

Преобразование ТД – см. операции, выражения

Базовые ТД и константы

```
Консоль отладки Microsoft Visual Studio
a=1256.780029 a=1256.78 b=222.550003 b=222.5500030518  c=1111.000000 dd=222.550003
a=1256.78 b=222.55 c=1111 dd=222.55
a=inf b=-nan(ind)
```

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

int main(){
    float a = (float)1256.78, b=222.55f, c=1111;        // Список переменных
    double dd=b;
    printf("a=%f a=%5.2f b=%f b=%12.10f  c=%f dd=%lf\n",a,a,b,b,c,dd);
    std::cout << "a=" << a << " b=" << b << " c=" <<  c <<  " dd=" << dd << "\n";
    a = a / 0;
    b = sqrt(-1);
    printf("a=%f b=%f\n", a, b);
}
```

- float (4 байта) – для хранения, double (8 байтов) – вычисления (выражения)
- float – мантисса = 24 разряда, 7 значащих цифр
- форматирование вывода – общее кол-во цифр и дробная часть
- ,бесконечность - ind, недопустимое - nan

Синтаксис определения переменных

```
int a, b = 55, fufu_666 = a + b;
```

- общий ТД слева
- список через запятую, в конце «;»
- имя = идентификаторы (буквы, цифры, «_», начинается с буквы)
- без = - не инициализирована, *глобальная* – по умолчанию, *локальная* – неопределенное (произвольное)
- = инициализатор, начальное значение, *локальная* – выражение, *глобальная* – константа (см. Функции. Виды переменных, область видимости, классы памяти)

Массивы

Массив - последовательность переменных одного и того же типа, имеющая общее имя. Номер элемента в последовательности называется **индексом**.

Свойства:

- количество элементов в массиве должно быть определено во время трансляции (задано константой) и не может быть изменено в процессе выполнения программы (статический массив)
- элементы массива размещаются в памяти последовательно и нумеруются от **0** до **n-1**, где **n** - их количество в массиве
- все элементы массива имеют одинаковую размерность, доступ к i-му элементу – вычисление адреса: нач.адрес + i*sizeof(тип элемента)

Массивы

Особенность массивов в Си: во время работы программы контроль за нахождением индексов в пределах размерности массива не производится. В случае выхода за пределы массива будут использованы значения переменных в соседних областях памяти и результат работы программы будет непредсказуем.

```
#include <iostream>
#include <stdio.h>

int x[100];                    // Неинициализированный – 100 элементов
int a[] = { 1,5,4,17 };        // Размерность выводится при компиляции из инициализатора
int b[] = { 1,2,3,4,5,6 };     // Размерность выводится при компиляции из инициализатора
int c[20] = { 1,5,4,7 };       // 20 элементов, инициализированы первые 4
int main(){
    int N = sizeof(b) / sizeof(int);// Количество элементов в b
    printf("b[2]=%d b[-1]=%d b[10]=%d\n", b[2], b[-1], b[10]);
}
```

```
Консоль отладки Microsoft Visual Studio
b[2]=3 b[-1]=17 b[10]=4
```

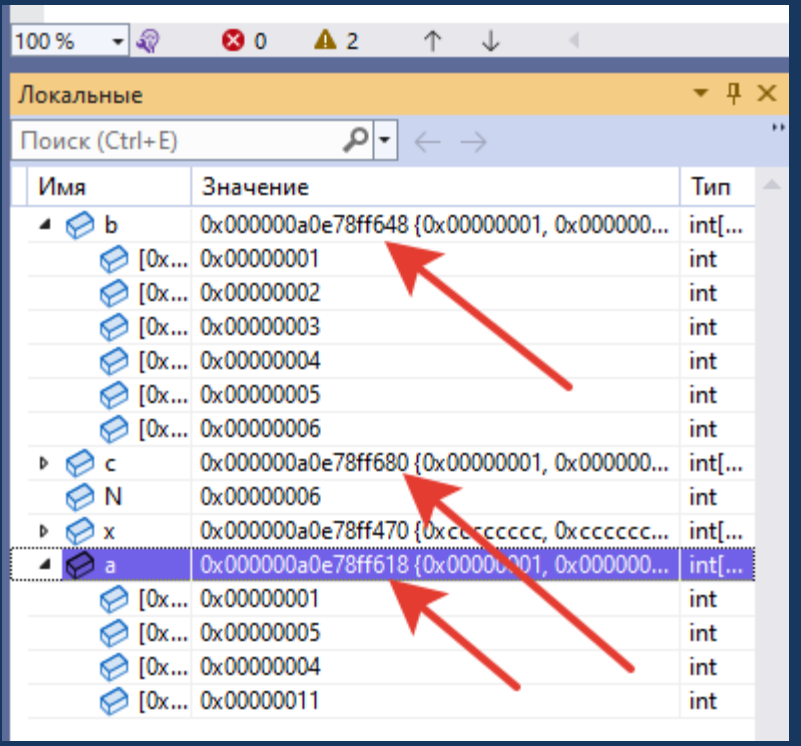
b[-1] == a[3] – последний
b[10] == c[2] – массив b выровнен до 8 элементов
Массивы *глобальные*

Массивы

```
F:\temp\ggg3\x64\Debug\ggg3.exe
b[2]=3 b[-9]=17 b[16]=4 N=6
```

```c
int main(){
    int x[100];                     // Неинициализированный – 100 элементов
    int a[] = { 1,5,4,17 };         // Размерность выводится при компиляции из инициализатора
    int b[] = { 1,2,3,4,5,6 };      // Размерность выводится при компиляции из инициализатора
    int c[20] = { 1,5,4,7 };        // 20 элементов, инициализированы первые 4
    int N = sizeof(b) / sizeof(int);// Количество элементов в b
    printf("b[2]=%d b[-9]=%d b[16]=%d N=%d\n", b[2], b[-9], b[16],N);
}
```

b[-9] == a[3]
b[16] == c[2]
Массивы *локальные*
Адрес a=0x…618
Адрес b=0x…648, размер a = 0x30= 48
Реальный размер int a[12]
Адрес c=0x…680, размер b = 0x38= 56
Реальный размер int a[14]



Контрольные вопросы

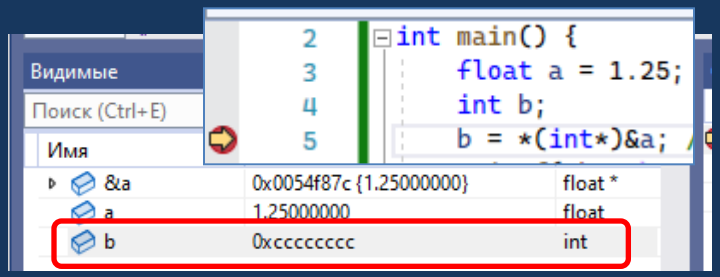
Для вывода значений в 16СС можно использовать спецификатор формата вывода в printf

```
#include <stdio.h>
void main() {
    int a = 0x1FF, b=258;
    printf("a=%d  %x\nb=%d %x", a, a, b, b);
    }
```

```
Консоль отла
a=511  1ff
b=258 102
```

Форматированный вывод [cprog 4.5]

1. Напишите константу в 16СС, содержащую чередующиеся двоичные разряды 01010101
2. Напишите константу в 16СС, содержащую чередующиеся триады двоичных разрядов 101
3. Напишите константу в 16СС, содержащую целое -130 в прямом и доп. коде
4. Возьмите день и месяц рождения, цифры 0 замените на 9, например 20.07 = 2997. Оцените количество значащих разрядов в числе, используя знания степеней 2
5. Возьмите день и месяц и год рождения, цифры 0 замените на 9, например 20.07.2003 = 29972993. Оцените количество значащих разрядов в числе, используя оценку 2^10 ≈ 10^3
6. Отладчик VStudio прописывает в неинициализированные переменные значение 0xCCCCCCCC. Какое значение из диапазона представления целых можно рекомендовать для этого случая?



Контрольные вопросы

7. Опишите поведение цикла (зависнет, завершится, «уронит» программу). Как это связано с форматом представления переменной этого типа?
   int a=1; while(a!=0) a++;
8. Опишите поведение цикла (зависнет, завершится, «уронит» программу). Как это связано с форматом представления переменной этого типа?
   int a=1; while(a>0) a++;