Лабораторная работа №3
Форматы данных в ЭВМ

Справочный материал

1. Представление данных в ЭВМ
2. Представление целых чисел
3. Представление вещественных чисел

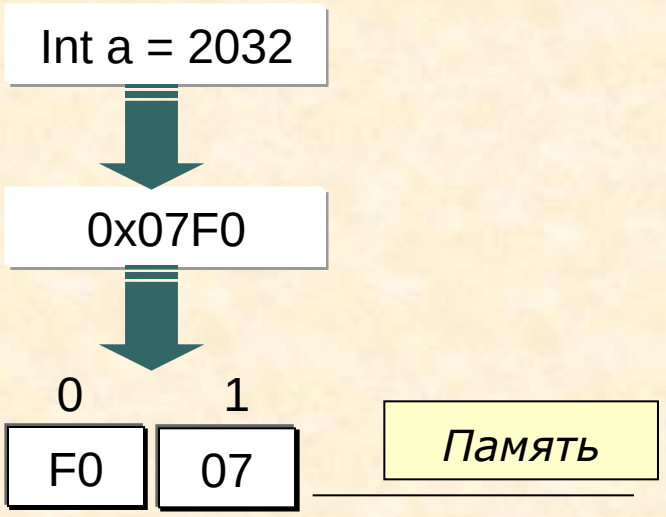
1

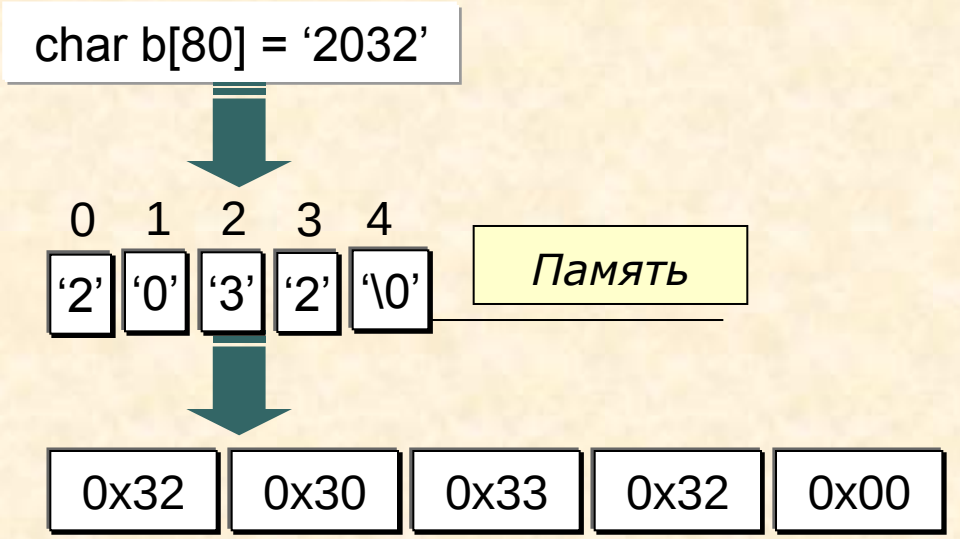
Представление данных в ЭВМ

Формы представления чисел

Внешняя форма представления числа. Представление числа в виде строки символов (цифр в заданной системе счисления).

Внутренняя форма представления числа. Представление числа в виде целой или вещественной переменной.





Базовые принципы

Базовый принцип арифметики

Положительное число, сложенное со своим отрицанием, в сумме дает ноль.

Базовый принцип двоичной арифметики

Дополнительный код положительного числа, сложенный с дополнительным кодом отрицания этого же числа, в сумме дает ноль.

Дополнительный код

	СЗБ	НДСС	ОК	ДК
$0\,1\,1\,1_2$	+7	+3	+7	+7
$1\,0\,1\,1_2$	-3	-9	-4	-5
$1\,0\,0\,1\,0_2$	+2	-2	+2	+2

Как придумали правило ДК? Почему нужно инвертировать биты и прибавлять 1?

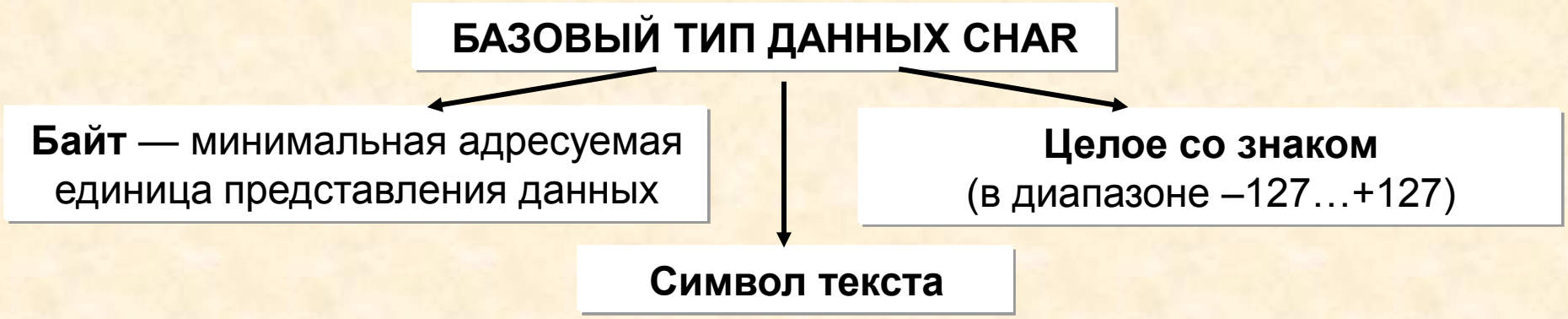
$x_{(2,n)} + inv(x_{(2,n)}) = ...11111111_{(2,n)} = 2^n - 1$. Пример: $1111_{(2,4)} = 2^4 - 1$

$inv(x_{(2,n)}) + 1 = 2^n - x_{(2,n)}$

$inv(x_{(2,n)}) + 1 = -x_{(2,n)}$

$a_{(2,n)} - b_{(2,n)} = a_{(2,n)} + (-b_{(2,n)}) = a_{(2,n)} + (2^n - b_{(2,n)}) = a_{(2,n)} + (inv(b_{(2,n)}) + 1)$

Интерпретация типа данных char



Тип данных char не имеет никаких ограничений на выполнение операций, допустимых для целых переменных.

Для представления отдельных символов можно пользоваться символьными (литерными) константами.
Транслятор вместо константы подставляет код соответствующего символа.

```
char b;
for (b = 'A'; b <= 'Z'; b++)…  ⇒  for (b = 0x41; b <= 0x5A; b++) ...
```

Сравнение символьных данных

Непрерывные фрагменты символьных кодировок

строчные буквы латинского алфавита
прописные буквы латинского алфавита
цифры

Знаки в этих фрагментах упорядочены по возрастанию.

Пример 1 Проверка принадлежности символа одному из интервалов (сравнение символа с концами интервала)

```
char a = ’f’;
if (a >= ’a’ && a <= ’z’)
printf(“true”);
```

Пример 2 Определение номера буквы в алфавите

```
char a = ’f’;
int num = a - ’a’;
printf(“%d”, num);
```

Посимвольная и пословная обработка

Логика переменных состояния
Программа содержит основной цикл посимвольного просмотра строки.
В зависимости от текущего символа меняются значения переменной состояния.
Пример. Выделение слов
Используется счетчик символов внутри слова, который обнуляется на каждом пробеле и увеличивается на 1 с каждым символом слова.
В момент обнуления счетчика программа находится в конце текущего слова.

Структурная логика
Устанавливается однозначное соответствие между элементами формата (повторениями) и управляющими конструкциями алгоритма (циклами).
Пример. Пословная обработка строки
Используется цикл, за каждый шаг которого просматривается одно слово (включая пробелы перед ним). Внутри него должны быть два цикла — по цепочке пробелов и по цепочке символов слова.

Символы и строки

Строка — последовательность символов, ограниченная символом '\0'.

Строка — объект переменной размерности.

Место хранения строки — массив символов.

Массив символов может быть инициализирован строкой или заполнен программно.

Пример

```
char A[20] = { 'C','т','р','о','к','а','\0' };
char B[40];
for (int i = 0; i < 20; i++) B[i] = 'A';
B[20] = '\0';
```

*Формат строки, ограниченной символом **'\0'**, используется для представления ее в памяти программы.*

*При чтении строки из внешнего потока (клавиатура, экран, файл) ограничителем строки является символ **'\n'**.*

Строковая константа

Строковая константа — последовательность символов, заключенная в двойные кавычки.

Строковая константа автоматически дополняется символом **'\0'**.

Строковой константой можно инициализироваться массив, в том числе такой, размерность которого определяется размерностью строки.

Пример

```
char a[80] = "123456\r\n";
char b[] = "aaaaa\033bbbb";
```

Представление текста

Текст является последовательностью строк.

Пример. Хранение текста в двумерном массиве символов

```
char a[20][80];
char b[][40] = { "Строка",«абвгд",«12345","abcdef"};
```

Первый индекс двумерного массива соответствует номеру строки, второй - номеру символа в нем.

Пример. Работа с символами i-й строки

```
for (int i = 0; i < 20; i++)
for (int j = 0; a[i][j] != '\0'; j++) {…}
```

Приемы обработки строк

ИЗМЕНЕНИЕ СОДЕРЖИМОГО СТРОКИ

Редактировать строку «на месте», реализуя вставку и удаление символов или фрагментов

Посимвольно переписывать входную строку в выходную, с копированием нужных фрагментов

Подсчет количества слов. Для программы необходимо формальное условие обнаружения слова. Это может быть или конец слова, или его начало.

Начало слова обнаруживается по сочетанию пары символов: текущий символ слова (не пробел), перед которым стоит либо пробел, либо начало строки.

Приемы обработки строк

Удаление лишних пробелов. Количество индексов определяет количество независимых перемещений по массивам (степеней свободы).

Для входной строки индекс может изменяться в заголовке цикла посимвольного просмотра.

Для выходной строки индекс меняется только в моменты добавления очередного символа.

Не забывайте «закрывать» выходную строку символом конца строки.

Сравнение строк. Сравнении кодов символов при наличии последовательного кодирования латинских букв и цифр дает гарантию их алфавитного упорядочения (цифры, прописные латинские, строчные латинские). Так работает стандартная функция **strcmp**.

Особенность работы со строками

Работать со строкой нужно в цикле, ограниченном не размерностью массива, а условием обнаружения символа конца строки —**'\0‘**.

```
for (i = 0; B[i] !='\0'; i++)...
```

Соответствие размерности массива и длины строки транслятором не контролируется. Это должна делать программа.

Пример Контроль переполнения массива

```
// Ограничение строки размерностью массива
char a[20], b[] =” Слишком длиииииииинная строка”;
for (i = 0; i < 19 && a[i] != '\0'; i++) b[i] = a[i];
b[i] = '\0';
```

2

Представление целых чисел

Представление целых чисел в ограниченной разрядной сетке

Пример (используется 32-разрядный компьютер)

Хранение переменной a = 2 в памяти компьютера

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{(2)}$.

Выполнение процессором операции сложения
(за один такт работы операция выполняется сразу со всеми 32-мя битами**)**

$$
\begin{array}{l}
+\ 00000000000000000000000000000010_{(2)} \\
\underline{00000000000000010000000000000010_{(2)}} \\
00000000000000010000000000000100_{(2)}
\end{array}
$$

Диапазон представления целых неотрицательных чисел

Пусть для хранения целого неотрицательного числа
в переменной **a** используется **k** бит.

MIN(a) = $000\ldots000_{(2)} = 0$ **MAX(a) = $111\ldots111_{(2)} = 2^k - 1$**

$$999 = 1000 - 1 = 10^3 - 1$$
$$111_{(2)} = 1000_{(2)} - 1 = 2^3 - 1$$

Диапазон представления целых неотрицательных чисел в k-разрядной сетке от 0 до $2^k - 1$.

Запись целых чисел со знаком

знак	модуль числа

Прямой код

$x_{пр} = x$, если $x >= 0$

$x_{пр} = 1 + x$, если $x <= 0$

$X^{+}_{мин} = 0|0\ldots0$ «+0»

$X^{+}_{макс} = 0|1\ldots1 = 1 - 2^{-n}$

$X^{-}_{мин} = 1|1\ldots1 = -(1 - 2^{-n})$

$X^{-}_{макс} = 1|0\ldots0$ «-0»

Дополнительный код

$x_{доп} = x$, если $x >= 0$

$x_{доп} = 10 + x$, если $x < 0$

$X^{+}_{мин} = 0,0\ldots0$ «+0»

$X^{+}_{макс} = 0,1\ldots1 = 1 - 2^{-n}$

$X^{-}_{мин} = 1,1\ldots1 = 2 - 2^{-n}$

$X^{-}_{макс} = 1,0\ldots0$

наибольшее по модулю отрицательное число

Обратный код

$x_{обр} = x$, если $x >= 0$

$x_{обр} = 1 + [(1 - 10_2^{-n}) - |x|] = 10_2 - (10_2)^{-n} - |x| = 10_2 - (10_2)^{-n} + x$, если $x <= 0$

Представление чисел с фиксированной точкой

У чисел с фиксированной точкой в двоичном формате предполагается строго определенное место точки (запятой).

Точка фиксируется перед первой значащей цифрой. Это означает, что число по модулю меньше единицы.

Диапазон изменения значений

$$2^{-n} \le |X| \le 1 - 2^{-n}$$

Точка фиксируется после последней значащей цифры. Это означает, что n-разрядные двоичные числа являются целыми.

Диапазон изменения значений

$$0 \le |X| \le 2^{n} - 1$$

Числа вне диапазона изменений

**Числа, которые выходят за диапазон изменения,
в ЭВМ не могут быть представлены точно.**

$|X| < |X|_{min} = 2^{-n}$, такое число воспринимается как нуль.

$|X| > |X|_{max} = 1 - 2^{-n}$, такое число воспринимается как бесконечно большое.

Этим двум случаям соответствуют понятия
машинный нуль и **машинная бесконечность.**

Точность представления чисел с фиксированной точкой

Абсолютная ошибка

$$|\Delta X| \le 0{,}5*2^{-n}$$

Минимальная относительная ошибка

$$|\delta x|_{min} = \frac{|\Delta X|}{|X|_{max}} = -\frac{0{,}5*2^{-n}}{1 - 2^{-n}} \approx 2^{-(n+1)},$$

Максимальная относительная ошибка

$$|\delta X|_{max} = \frac{|\Delta X|}{|X|_{min}} = \frac{0{,}5*2^{-n}}{2^{-n}} = 0{,}5$$

Ошибка представления числа зависит от величины самого числа и способа округления

$$2^{-(n+1)} \le |\delta X| \le 0{,}5$$

3

Представление вещественных чисел

Параметры двоичных форматов представления чисел согласно IEEE 754

Параметр (бит)	Binary32	Binary64	Binary128
Разрядная сетка	32	64	
Точность	24	53	
Max показатель степени	127	1023	
Смещение	127	1023	
Знак	1	1	
Порядок	8	11	
Мантисса	23	52	

Заполнить таблицу

?

Международные стандартные форматы различаются по точности, но имеют одинаковую структуру

Знак мантиссы	Смещенный порядок (характеристика)	Абсолютная величина мантиссы

Вычисление характеристики

Тип	Харрактеристика	Кол-во бит на хар-ку
real	x = 2^7 + p + 1	8
single	x = 2^7 + p - 1	8
double	x = 2^10 + p - 1	11
extended	x = 2^14 + p - 1	15

Нормализация мантиссы

Для увеличения количества значащих цифр в представлении числа и для исключения переполнения при умножении мантиссу подвергают нормализации.

Нормализация означает, что мантисса M, кроме случая, когда M = 0, должна находиться в интервале $\mathbf{P^{-1} <= M < 1.}$

Для двоичной системы счисления P = 2.

Так как $2^{-1} <= M < 1$, то ненулевая мантисса любого хранимого числа с плавающей точкой должна начинаться с двоичной единицы.

Процесс нормализации создает дробь, первый бит которой равен 1. Эта единица учитывается, но не записывается в мантиссу (скрытая единица). Получающийся дополнительный бит используют для увеличения точности представления чисел или их диапазона.

Условие нормализации для мантиссы

$1 <= |M_{\text{фактическая 10}}| < p$, p — основание системы счисления.

$1_{10} <= |M_{\text{фактическая 10}}| < 2_{10}$ или $1_2 <= |M_{\text{фактическая 2}}| < 10_2$

$1{,}000000\ldots_2 <= |M_{\text{фактическая 2}}| < 10{,}000000\ldots_2$

Реальная мантисса есть число вида 1,xxxxxxxx…

Знак мантиссы	Смещенный порядок (характеристика)	Абсолютная величина мантиссы (без скрытой 1)

Диапазон мантиссы денормализованных чисел

$0_2 < |M_{\text{фактическая 2}}| < 1_2$

Диапазон и точность представления вещественных чисел

Диапазон изменения

$$2^{-2^k} \le |X| < (1-2^{-m})*2^{+(2^k-1)}$$

Абсолютная ошибка

$$|\Delta X| \le 0{,}5*2^{-m}$$

Так как $2^{-1} \le |M_x| \le 1 - 2^{-m}$, то

минимальная относительная ошибка

$|\Delta X|_{min} = (0{,}5*2^{-m}) / (1 - 2^{-m}) \approx 2^{-(m+1)}$, при m — большом.

Максимальная относительная ошибка

$|\Delta X|_{max} = (0{,}5*2^{-m}) / (2^{-1}) = 2^{-m}$

Особые значения вещественных чисел

Порядок, равный 255 при нулевой мантиссе, означает **бесконечность**.

Порядок, равный 255 при ненулевой мантиссе, представляет значение NaN — Not-a-Number — **не число**.
Возникает при выполнении недопустимой операции типа 0 / 0, sqrt(-x).

Если порядок равен нулю, а мантисса не равна нулю, то число называется **анормальным** (субнормальным, денормализованным).
Может быть использовано для представления положительных и отрицательных нулей, а также значений, меньших минимального нормализованного числа.

Если порядок и мантисса равны нулю, то число считается **нулем**.

Стандартные форматы представления вещественного числа

Одинарный (диапазон $1{,}17*10^{-38} – 1{,}7*10^{38}$)
Нормализованное число со знаком — 32 разряда.
Смещенный порядок — 8 разрядов.
Мантисса — 23 разряда

Двойной ($2{,}22*10^{-308} – 1{,}79*10^{308}$)
Нормализованное число со знаком — 64 разряда.
Смещенный порядок — 11 разрядов.
Мантисса — 52 разряда.

Расширенный ($3{,}4*10^{-4932} – 1{,}2*10^{4932}$)
Число со знаком — 80 разрядов.
Смещенный порядок — 15 разрядов.
Мантисса — 64 разряда.
Позволяет хранить ненормализованные числа.

Стандарты для машинной арифметики с плавающей точкой

IEEE 754-2008 Standard for Floating-Point Arithmetic

Рассматривает

форматы хранения
правила арифметики
правила округления
форматы для обмена данными

Определяет как представлять

нормализованные «+» и «–» числа
денормализованные «+» и «–» числа
«нулевые» числа
специальные величины «+∞» и «-∞»
специальные величины «не число»

IEEE 854-1987 Standard for Radix-Independent Floating-Point Arithmetic

Обобщает стандарт ANSI/IEEE Std 754-1985

Цель: убрать зависимость от основания системы счисления и длины машинного слова.

Типы данных в языке C++

«… типы с плавающей точкой представлены тремя размерами:
float (одинарной точности),
double (двойной точности),
long double(расширенной точности).
… Выбор нужной точности в реальных задачах требует хорошего понимания природы машинных вычислений с плавающей точкой»

(Бьярне Страуструп, автор языка C++)

Документ draft ISO/IEC JTC1 SC22 WG14 N1312 определяет типы float, double, long double как базовые для языка C++

Вычисления с плавающей точкой

Формат СПТ — форма представления действительных чисел, в которой число хранится в виде мантиссы и показателя степени.

Число с плавающей точкой имеет фиксированную относительную точность и изменяющуюся абсолютную.

Тип данных	Байты	Биты	Мантисса	Диапазон	Формат ввода-вывода
Float	4	32	7 десятичных цифр	$1.5\ 10^{-45}$ .. $3.4\ 10^{38}$	`%f`
double	8	64	15 десятичных цифр	$5.0\ 10^{-324}$ .. $1.7\ 10^{308}$	`%lf`
long double	10	80		$3.4\ 10^{-4932}$ .. $1.1\ 10^{4932}$	`%Lf` или `%llf`

Особенности формата СПТ

Мантисса двоичного числа **1 <= M < 2**.

Первый бит всегда равен 1 и не хранится. В такой форме любое число записывается единственным образом.

Недостаток: невозможно представить 0.

Арифметические операции с плавающей точкой занимают значительно больше времени, чем с целыми числами.

Дополнительный процессор для вычислений с плавающей точкой использует специальные регистры.

Размер регистров — 80 бит.

Вычисления с плавающей точкой всегда производятся в типе **long double**.

Пример представления чисел в формате IEEE 754

Любое двоичное целое число, содержащее не более m разрядов, может быть без искажений преобразовано в вещественный формат с m-разрядной мантиссой.

Десятичное число **36,6** в формате IEEE 754 с одинарной точностью

66 66 12 42 (hex)

Десятичное число **36,6** в памяти ЭВМ с архитектурой LITTLE-ENDIAN

42 12 66 66 (hex)

Онлайн-преобразователь для IEEE 754 чисел с двойной точностью

www.binaryconvert.com/convert-double.html