Основы программирования и баз данных

Модуль 2.
ПРЕДСТАВЛЕНИЕ ДАННЫХ. ПРИНЦИП ПРОГРАММНОГО УПРАВЛЕНИЯ

- Основы булевой алгебры
- Системы счисления. Связи между системами счисления;
- Основы арифметики двоичных чисел
- Принцип программного управления.
  Базовая архитектура и структура ЭВМ.
  Принцип фон Неймана

Модуль 2.
ПРЕДСТАВЛЕНИЕ ДАННЫХ. ПРИНЦИП ПРОГРАММНОГО УПРАВЛЕНИЯ
(продолжение)

- Единицы измерения ёмкости запоминающих устройств
- Представление целых и вещественных чисел в памяти ЭВМ;
- Диапазоны представления чисел в двоичной системе счисления;
- Представление символьной информации. Кодовые таблицы;
- Понятие типа данных.

Основы булевой алгебры

- **Булевый** (логический) тип данных — в информатике является примитивным типом данных имеющим два возможных значения:
  - ***true*** **(*правда*)**
  - ***false*** **(*ложь*)**

- Присутствует в подавляющем большинстве языков программирования как самостоятельная сущность или реализуется через численный тип. В подавляющем большинстве языков значение ***true*** представляется единицей, а ***false*** - нулем

Материал из Википедии — свободной энциклопедии

Основы булевой алгебры (продолжение)

- Традиционным применением булевого типа данных являются значения «да»/«нет» в отношении результата более сложных операций.
- Все операции сравнения двух величин (равно, больше, меньше), операции вхождения элемента в множество и проверка на пересечение множеств возвращают в качестве результата булевый тип.

Материал из Википедии — свободной энциклопедии

Основы булевой алгебры (продолжение)

- К логическому типу данных чаще всего применяют следующие операции:
  - ***эквивалентность*** (равенство) (EQV, =, ==)
  - ***отрицание*** (инверсия) (NOT, ~, !)
  - ***конъюнкция*** (***И,*** логическое умножение) (AND, &, *)
  - ***дизъюнкция*** (***ИЛИ***, логическое сложение) (OR, |, +),
  - ***исключающее ИЛИ*** (сложение по модулю 2) (NEQV, XOR, ^)
- Также существуют и другие операции булевой алгебры

Материал из Википедии — свободной энциклопедии

Основы булевой алгебры (продолжение)

Таблица истинности унарных операций

x	false	EQ	NOT	true
0	0	0	1	1
1	0	1	0	1

Таблица истинности бинарных операций

x	y	false	AND			XOR	OR			true
0	0	0	0	0	…	0	0	1	..	1
0	1	0	0	0	…	1	1	0	…	1
1	0	0	0	1	…	1	1	0	…	1
1	1	0	1	0	…	0	1	0	…	1

Основы булевой алгебры (продолжение)

- Например: при a=1, b=3, c = 5, d = 6
  - a + c = d возвращает true,
  - c = b возвращает false,
  - NOT(a + c = d) возвращает false,
  - NOT(c = b) возвращает true,
  - a < b AND b < c возвращает true,
  - a < d AND d < c возвращает false,
  - a < d OR d < c возвращает true,
  - d < a OR d < c возвращает false.
- Для упрощения сложных логических выражений применяют эквивалентные преобразования (**законы де Моргана)**:
  - (x AND y) OR (x AND z) = x AND (y OR z)
  - (x OR y) AND (x OR z) = x OR (y AND z)

Системы счисления. Связи между системами счисления

- **Система счисления** — способ записи чисел с помощью набора специальных знаков, называемых *цифрами*.

- Системы счисления подразделяются на
  - **позиционные** (например, десятичная)
  - **непозиционные** (например, римская)

- В позиционных системах счисления величина, обозначаемая цифрой в записи числа, зависит от её положения в числе (позиции).
- Количество используемых цифр называется ***основанием системы счисления***

Материал из Википедии — свободной энциклопедии

Системы счисления. Связи между системами счисления (продолжение)

- **Определение:**

***b*-ичная** система счисления определяется натуральным числом $b > 1$, называемым *основанием системы счисления*.

Для представления числа x в b-ичной системе счисления его представляют в виде линейной комбинации степеней числа b:

$$a_n b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \ldots + a_2 b^2 + a_1 b^1 + a_0 b^0,$$

где каждая b-ичная цифра удовлетворяет условию
$0 \le a_k < b$.

Материал из Википедии — свободной энциклопедии

Системы счисления. Связи между системами счисления (продолжение)

1) десятичная

10^3	10^2	10^1	10^0	Значение
1	1	0	9	$1*10^3+1*10^2+0*10^1+9*10^0=1109_{(10)}$

2) двоичная

2^3	2^2	2^1	2^0	Значение
1	1	0	1	$1*2^3+1*2^2+0*2^1+1*2^0=13_{(10)}$

3) восьмеричная

8^3	8^2	8^1	8^0	Значение
1	1	0	7	$1*8^3+1*8^2+0*8^1+7*8^0=583_{(10)}$

4) шестнадцатеричная

16^3	16^2	16^1	16^0	Значение
1	1	0	F	$1*16^3+1*16^2+0*16^1+15*16^0=4367_{(10)}$

Системы счисления. Связи между системами счисления (продолжение)

- **Перевод произвольной позиционной системы счисления в десятичную:**

  Если число в *b*-ичной системе счисления имеет запись
  $a_n\, a_{n-1}\, a_{n-2} \ldots a_2\, a_1\, a_0$
  то для перевода в десятичную систему вычисляем такую сумму:
  $a_n b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \ldots + a_2 b^2 + a_1 b^1 + a_0 b^0$

- **Пример:**

  $101100_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$
  $= 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1$
  $= 32 + 8 + 4$
  $= 44_{10}$

Материал из Википедии — свободной энциклопедии

Системы счисления. Связи между системами счисления (продолжение)

- **Перевод из десятичной в произвольную позиционную систему счисления:**
  Для перевода необходимо делить с остатком искомое число на основание системы счисления до тех пор, пока частное больше нуля, и записать цифры всех остатков в обратном порядке.
- **Пример:**
  - **44_{10}** переведём в двоичную систему:
  - 44 делим на 2. частное 22, остаток 0
  - 22 делим на 2. частное 11, остаток 0
  - 11 делим на 2. частное 5, остаток 1
  - 5 делим на 2. частное 2, остаток 1
  - 2 делим на 2. частное 1, остаток 0
  - 1 делим на 2. частное 0, остаток 1
- Теперь, записав цифры всех остатков в обратном порядке, получим число **101100_2**

Материал из Википедии — свободной энциклопедии

Системы счисления. Связи между системами счисления (продолжение)

- **Перевод из двоичной в восьмеричную и шестнадцатеричную системы**

  Для этого типа операций существует упрощенный алгоритм.
- Для восьмеричной — разбиваем число на триады, преобразуем триады по таблице
- Для шестнадцатеричной — разбиваем число на тетрады, преобразуем тетрады по таблице
- Пример:
  - преобразуем $\mathbf{101100_2}$
  - восьмеричная — 101 100 → $\mathbf{54_8}$
  - шестнадцатеричная — 0010 1100 → $\mathbf{2C_{16}}$

0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	A
1 0 1 1	B
1 1 0 0	C
1 1 0 1	D
1 1 1 0	E
1 1 1 1	F

Материал из Википедии — свободной энциклопедии

Системы счисления. Связи между системами счисления (продолжение)

- **Перевод из восьмеричной и шестнадцатеричной систем в двоичную**

  Для этого типа операций тоже существует упрощенный алгоритм.
- Для восьмеричной — преобразуем цифры числа по таблице в триады
- Для шестнадцатеричной — преобразуем цифры числа по таблице в тетрады
- Пример:
  - преобразуем
  - $54_8 \rightarrow$ 101 100
  - $2C_{16} \rightarrow$ 0010 1100

0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	A
1 0 1 1	B
1 1 0 0	C
1 1 0 1	D
1 1 1 0	E
1 1 1 1	F

Материал из Википедии — свободной энциклопедии

Основы арифметики двоичных чисел

Поразрядное сложение с переносом

0	1	1	1	= ?
0	1	1	0	= ?
?	?	?	?	= ?

0	1	1	1	= 7
0	1	1	0	= 6
1	1	0	1	=13

Сдвиг влево

0	0	1	1	= ?
0	1	1	0	= ?
1	1	0	0	= ?

0	0	1	1	= 3
0	1	1	0	= 6
1	1	0	0	=12

Сдвиг вправо

1	1	0	1	= ?
0	1	1	0	= ?
0	0	1	1	= ?

1	1	0	1	= 13
0	1	1	0	= 6
0	0	1	1	= 3

2^3	2^2	2^1	2^0	Значение$_{(10)}$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Принцип программного управления.

- Принцип программного управления:
    - функционирование вычислительной машины определяется заранее составленной и введенной в ее память программой
    - команды программы располагаются в последовательных адресах памяти
    - после исполнения первой команды машина автоматически переходит к выполнению следующей команды и т.д., пока не встретится команда прекратить вычисления
- Команда содержит
    - код операции (сложить, умножить, записать в память, перейти по адресу и т.п.)
    - адреса одного или нескольких операндов или (реже) их значения, а для команды перехода - адрес следующей команды

Базовая архитектура и структура ЭВМ. Принцип фон Неймана

- Первые компьютеры использовали запоминающие устройства исключительно для хранения обрабатываемых данных.
- Их программы реализовывались на аппаратном уровне (при помощи коммутирующих перемычек) в виде жёстко заданных выполняемых последовательностей.
- Любое перепрограммирование требовало огромного объёма ручной работы по подготовке новой документации, перекоммутации, перестройке блоков и устройств и т. п.
- Использование новой архитектуры, которая, согласно принципу фон Неймана, предусматривает хранение компьютерных программ и данных в общей памяти, коренным образом изменило ситуацию.

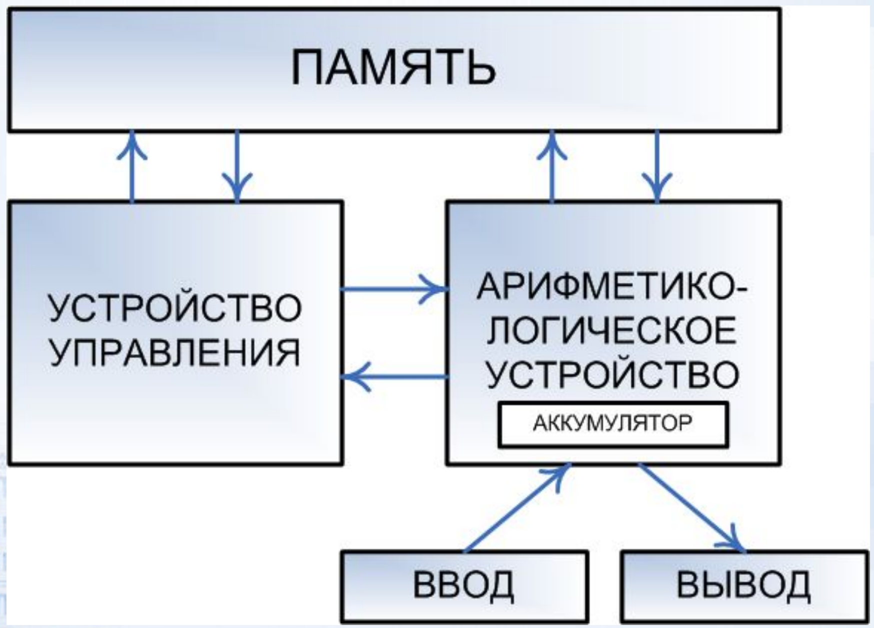
Материал из Википедии — свободной энциклопедии

Базовая архитектура и структура ЭВМ. Принцип фон Неймана (продолжение)

- **Машина фон Неймана** — вычислительная система, построенная на следующих принципах.
  - Основными ее блоками являются:
    - арифметико-логическое устройство,
    - устройство управления,
    - запоминающее устройство,
    - устройства ввода-вывода.
  - Программы и данные хранятся в одной и той же памяти.
  - Устройство управления и арифметико-логическое устройство, объединенные в центральный процессор, определяют действия, подлежащие выполнению, путем считывания команд из оперативной памяти.
- Подавляющее большинство вычислительных машин в настоящее время являются фон-неймановскими машинами.

Материал из Википедии — свободной энциклопедии

Базовая архитектура и структура ЭВМ. Принцип фон Неймана (продолжение)



Схематичное изображение машины фон Неймана

Материал из Википедии — свободной энциклопедии

Единицы измерения ёмкости запоминающих устройств

- 1 бит = двоичная цифра /логическое значение
- 8 бит = 1 байт - символ (ASCII)
- 1 Кб = 1024б = 2^{10} байт - килобайт
- 1 Мб = 1024Кб = 2^{20} байт - мегабайт
- 1 Гб = 1024Мб = 2^{30} байт - гигабайт
- 1 Тб = 1024Гб = 2^{40} байт - терабайт

Представление целых и вещественных чисел в памяти ЭВМ

Поразрядное сложение с переносом

0	0	1	0	= ?
1	0	1	1	= ?
?	?	?	?	= ?

	0	0	1	0	= 2
	1	0	1	1	=11
	1	1	0	1	=13

1	1	1	1	= ?
0	0	0	1	= ?
?	?	?	?	= ?

	1	1	1	1	=15
	0	0	0	1	= 1
~~1~~	0	0	0	0	= 0

1	1	1	0	= ?
0	1	0	1	= ?
?	?	?	?	= ?

	1	1	1	0	=14
	0	1	0	1	= 5
~~1~~	0	0	1	1	= 3

Сдвиг влево (умножение на 2)

0	1	1	0	= ?
1	1	0	0	= ?
?	?	?	?	= ?

	0	1	1	0	= 6
	1	1	0	0	=12
~~1~~	1	0	0	0	= 8

Переполнение разрядной сетки

2^3	2^2	2^1	2^0	Значение$_{(10)}$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Представление целых и вещественных чисел в памяти ЭВМ (продолжение)

Отрицательные числа: -1, -5 и т.д.

Определение: x + (-x) = 0

0	0	0	1	= 1
?	?	?	?	= -1
0	0	0	0	= 0

	0	0	0	1	= 1
	1	1	1	1	= -1
~~1~~	0	0	0	0	= 0

0	1	0	1	= 5
?	?	?	?	= -5
0	0	0	0	= 0

	0	1	0	1	= 5
	1	0	1	1	= -5
~~1~~	0	0	0	0	= 0

Изменение знака числа:
заменить все 0 на 1, а 1 - на 0 (NOT)
и к результату прибавить 1

0	1	0	1	= 5
1	0	1	0	NOT
0	0	0	1	+1
1	0	1	1	= -5

1	0	1	1	= -5
0	1	0	0	NOT
0	0	0	1	+1
0	1	0	1	= 5

±	2^2	2^1	2^0	Значение (10)	Со знаком (10)
0	0	0	0	+0	+0
0	0	0	1	+1	+1
0	0	1	0	+2	+2
0	0	1	1	+3	+3
0	1	0	0	+4	+4
0	1	0	1	+5	+5
0	1	1	0	+6	+6
0	1	1	1	+7	+7
1	0	0	0	- ?	- 8
1	0	0	1	- ?	- 7
1	0	1	0	- ?	- 6
1	0	1	1	- ?	- 5
1	1	0	0	- ?	- 4
1	1	0	1	- ?	- 3
1	1	1	0	- ?	- 2
1	1	1	1	- ?	- 1

Представление целых и вещественных чисел в памяти ЭВМ (продолжение)

Суммирование чисел со знаком:

0	0	1	0	= ?		0	0	1	0	= 2
0	1	0	1	= ?		0	1	0	1	= 5
?	?	?	?	= ?		0	1	1	1	= 7

0	0	1	0	= ?		0	0	1	0	= 2
1	0	1	1	= ?		1	0	1	1	= -5
?	?	?	?	= ?		1	1	0	1	= -3

1	1	1	0	= ?		1	1	1	0	= -2
0	1	0	1	= ?		0	1	0	1	= 5
?	?	?	?	= ?	~~1~~	0	0	1	1	= 3

1	1	1	0	= ?		1	1	1	0	= -2
1	0	1	1	= ?		1	0	1	1	= -5
?	?	?	?	= ?	~~1~~	1	0	0	1	= -7

±	2^2	2^1	2^0	Со знаком$_{(10)}$
0	0	0	0	+0
0	0	0	1	+1
0	0	1	0	+2
0	0	1	1	+3
0	1	0	0	+4
0	1	0	1	+5
0	1	1	0	+6
0	1	1	1	+7
1	0	0	0	- 8
1	0	0	1	- 7
1	0	1	0	- 6
1	0	1	1	- 5
1	1	0	0	- 4
1	1	0	1	- 3
1	1	1	0	- 2
1	1	1	1	- 1

Представление целых и вещественных чисел в памяти ЭВМ (продолжение)

Сдвиг вправо (деление на 2)

0	1	1	0	= 6
0	0	1	1	= 3
0	0	0	1	= 1

1	0	1	0	=-6
1	1	0	1	=-3
1	1	1	0	=-2

Проблемы со сложением

1	0	1	1	= ?
1	0	1	1	= ?
?	?	?	?	= ?

	1	0	1	1	= -5
	1	0	1	1	= -5
~~1~~	0	1	1	0	=+6

0	1	0	1	= ?
0	1	0	1	= ?
?	?	?	?	= ?

0	1	0	1	= 5
0	1	0	1	= 5
1	0	1	0	= -6

Проблемы со сдвигом влево (умн. на 2)

0	1	0	1	= ?
1	0	1	0	= ?
?	?	?	?	= ?

	0	1	0	1	=+5
	1	0	1	0	= -6
~~1~~	0	1	0	0	=+4

±	2^2	2^1	2^0	Со знаком$_{(10)}$
0	0	0	0	+0
0	0	0	1	+1
0	0	1	0	+2
0	0	1	1	+3
0	1	0	0	+4
0	1	0	1	+5
0	1	1	0	+6
0	1	1	1	+7
1	0	0	0	- 8
1	0	0	1	- 7
1	0	1	0	- 6
1	0	1	1	- 5
1	1	0	0	- 4
1	1	0	1	- 3
1	1	1	0	- 2
1	1	1	1	- 1

Представление целых и вещественных чисел в памяти ЭВМ (продолжение)

- Для представления дробной части числа в *b*-ичной системе счисления ее представляют в виде линейной комбинации отрицательных степеней числа *b*:

$$a_2 b^2 + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \dots,$$

- Например, в двоичной системе (b = 2) в формате с фиксированной точкой имеем:

$0.1 = 0.50_{(10)}$

$0.01 = 0.25_{(10)}$

$0.11 = 0.75_{(10)}$

$1.11 = 1.75_{(10)}$

±	2^0	2^{-1}	2^{-2}	Значение$_{(10)}$
0	0	0	0	+0.00
0	0	0	1	+0.25
0	0	1	0	+0.50
0	0	1	1	+0.75
0	1	0	0	+1.00
0	1	0	1	+1.25
0	1	1	0	+1.50
0	1	1	1	+1.75
1	0	0	0	NAN
1	0	0	1	-0.25
1	0	1	0	-0.50
1	0	1	1	-0.75
1	1	0	0	-1.00
1	1	0	1	-1.25
1	1	1	0	-1.50
1	1	1	1	-1.75

Представление целых и вещественных чисел в памяти ЭВМ (продолжение)

- В формате с плавающей точкой (экспоненциальный формат) возможно многими способами записать одно и то же число:

  - в десятичной системе:

  $$3.14 * 10^{0} = 31.4 * 10^{-1} = 314 * 10^{-2} = 0.314 * 10^{1} = 0.0314 * 10^{2} \text{ и т.д.}$$

  - в двоичной системе:

  $$1.01 * 2^{0} = 10.1 * 2^{-1} = 101 * 2^{-2} = 0.101 * 2^{1} = 0.0101 * 2^{2} \text{ и т.д.}$$

- В памяти компьютера вещественные числа хранятся в нормализованной форме с плавающей точкой (стандарт IEEE):

  $$\pm 1.m * 2^{\pm p}$$

  где **m** - мантисса (дробная часть) числа, а **p** - порядок.

±	p	m

причем, **1** (целая часть числа) не записывается, но подразумевается, а порядок **p** хранится в смещенном формате

Диапазоны представления чисел в двоичной системе счисления

Целые числа

без знака:

от **0** до $\mathbf{2^n - 1}$

со знаком:

от $\mathbf{-2^{n-1}}$ до $\mathbf{2^{n-1} - 1}$

2^3	2^2	2^1	2^0	Без знака	Со знаком
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	-8
1	0	0	1	9	-7
1	0	1	0	10	-6
1	0	1	1	11	-5
1	1	0	0	12	-4
1	1	0	1	13	-3
1	1	1	0	14	-2
1	1	1	1	15	-1

Диапазоны представления чисел в двоичной системе счисления (продолжение)

- Числа с плавающей точкой имеют два формата:
  - при хранении с одинарной (обычной) точностью
    - используется **32** бита
    - обеспечивается диапазон значений от $\mathbf{10^{-38}}$ до $\mathbf{10^{+38}}$
    - обеспечивается точность **6** верных десятичных цифр

±	p (7 бит)	m (24 бита)

  - при хранении с удвоенной точностью
    - используется **64** бита
    - обеспечивается диапазон значений от $\mathbf{10^{-308}}$ до $\mathbf{10^{+308}}$
    - обеспечивается точность **15** верных десятичных цифр

±	p (10 бит)	m (53 бита)

Представление символьной информации. Кодовые таблицы

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		►		0	@	P	`	p	А	Р	а	░	└	╨	р	Ё
1	☺	◄	!	1	A	Q	a	q	Б	С	б	▒	┴	╤	с	ё
2	☻	↕	"	2	B	R	b	r	В	Т	в	▓	┬	╥	т	Є
3	♥	‼	#	3	C	S	c	s	Г	У	г	│	├	╙	у	є
4	♦	¶	$	4	D	T	d	t	Д	Ф	д	┤	─	╘	ф	Ї
5	♣	§	%	5	E	U	e	u	Е	Х	е	╡	┼	╒	х	ї
6	♠	▬	&	6	F	V	f	v	Ж	Ц	ж	╢	╞	╓	ц	Ў
7		↨	'	7	G	W	g	w	З	Ч	з	╖	╟	╫	ч	ў
8		↑	(	8	H	X	h	x	И	Ш	и	╕	╚	╪	ш	°
9		↓	)	9	I	Y	i	y	Й	Щ	й	╣	╔	┘	щ	·
A		→	*	:	J	Z	j	z	К	Ъ	к	║	╩	┌	ъ	·
B	♂	←	+	;	K	[	k	{	Л	Ы	л	╗	╦	█	ы	√
C	♀	∟	,	<	L	\	l	|	М	Ь	м	╝	╠	▄	ь	№
D		↔	-	=	M	]	m	}	Н	Э	н	╜	═	▌	э	¤
E	♬	▲	.	>	N	^	n	~	О	Ю	о	╛	╬	▐	ю	■
F	☼	▼	/	?	O	_	o	⌂	П	Я	п	┐	╧	▀	я	

- Однобайтные таблицы (ASCII, ANSI, KOI-8R)
  - для представления символов используются 8-битные числовые коды
  - кодовая таблица позволяет закодировать 256 различных символов
- Двухбайтная таблица (UNICODE)
  - для представления символов используются 16-битные числовые коды
  - кодовая таблица позволяет закодировать 65536 различных символов

Понятие типа данных

Тип данных определяет:

- объем *блока памяти*, выделяемый для хранения значений:
  - 1 байт для символьного типа
  - 4 байта для целого типа
- *структурную организацию* этого блока памяти:
  - наличие или отсутствие знакового разряда для целого типа
  - наличие знакового разряда, полей порядка и мантиссы для плавающего типа
- *диапазон* возможных значений:
  - от 0 до 255 (от 00 до FF) для символьного типа
  - от -2^{n-1} до $2^{n-1}-1$ для целого типа
- набор возможных *операций*, применяемых к этим значениям:
  - для значений плавающего типа не определена операция вычисления остатка от деления
  - к значениям логического типа применяются операции отрицания, конъюнкции, дизъюнкции

Понятие типа данных (продолжение)

В различных языках программирования реализованы те или иные из перечисленных ниже типов:

- простые (скалярные) типы:
  - логический
  - символьный
  - целый
  - с плавающей точкой
  - строковый
  - перечислимый
  - ссылочный (указатель)
- составные (структурные) типы:
  - массивы
  - записи (структуры)
  - множества
  - списки
- другие типы, определяемые программистом

Понятие типа данных (продолжение)

- Вообще говоря, в памяти компьютера хранятся только *последовательности битов*.
  - Если *имя переменной* указывает *адрес* в памяти, по которому хранится информация,
  - то *тип данных* (тип переменной) сообщает, каким образом следует обращаться с этой информацией.
- *Контроль типов*, т.е. проверка правильности и допустимости использования типов, может выполняться:
  - во время компиляции программы (*статическая* проверка)
  - во время выполнения программы (*динамическая* проверка).
- Статический контроль типов является основной задачей *семантического анализа*, выполняемого компилятором.
- Контроль типов может быть *сильным* и *слабым*.

Материал из Википедии — свободной энциклопедии

Понятие типа данных (продолжение)

- Преимуществом использования типов данных является *надёжность*.
- Типы данных защищают от трёх видов ошибок:
  - *Некорректное присваивание*.
    - Попытка присвоить числовой переменной строковое или другое недопустимое значение приведет к ошибке при контроле типов и позволит избежать многих трудностей.
  - *Некорректная операция*.
    - Контроль типов позволяет избежать попыток применения выражений вида «Hello world» + 1. Поскольку переменные в памяти хранятся как наборы битов, то при отсутствии контроля подобная операция была возможна и могла бы дать результат вроде «Hello worle».
  - *Некорректная передача параметров в функцию*.
    - Если функция «синус» ожидает, что ей будет передан числовой аргумент, то передача ей в качестве параметра строки «Hello world» (без контроля типов) может иметь непредсказуемые последствия.

Материал из Википедии — свободной энциклопедии