Структуры и алгоритмы обработки данных

Лекция 3

Базовые типы данных языков программирования высокого уровня (ч.1)

Данные - любой набор знаков, рассматриваемый безотносительно к его содержательному смыслу

ЭВМ в настоящее время:

- *считывает и выполняет определенные алгоритмы*
- *хранит значительные объемы информации, к которой нужно быстро обращаться*

Эта информация - абстракция фрагмента реального мира, состоит из определенного множества данных, относящихся к какой-либо проблеме

Данные изображают некоторую информацию, которую можно получить, если известен смысл, приписываемый данным

В программировании часто приходится иметь дело именно с данными

Например, разработка системы хранения и поиска некоторых текстов

Данные - любой набор знаков, рассматриваемый безотносительно к его содержательному смыслу

Вычислительные машины выполняют только обработку данных, которая заинтересованным лицам, приписывающим этим данным некоторый смысл, представляется обработкой информации

Пример

Программист разрабатывает систему хранения и поиска текстов

Задача - обеспечить экономное использование памяти и быстрый поиск требуемых текстов по заданным признакам

Достаточно знать лишь количественные характеристики текстов, рассматриваемых как данные

Программист может не знать содержания текстов!

Понятие структуры данных

Совокупности данных, организованные некоторым образом, называют структурами данных

Структура определяется отношениями между ее элементами

УРОВНИ ПРЕДСТАВЛЕНИЯ СТРУКТУР ДАННЫХ:

- **абстрактный (математический) уровень**
- **логический уровень**
- **физический уровень**

Понятие структуры данных

УРОВНИ ПРЕДСТАВЛЕНИЯ СТРУКТУР ДАННЫХ:

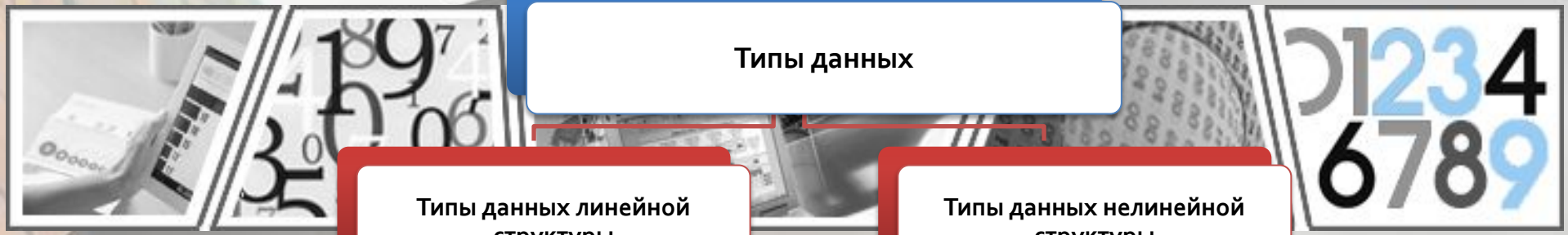
- **Физический уровень –**
  отображение на память ЭВМ информационного объекта в соответствии с логическим описанием

На этом уровне определяются

- *область и объём памяти, необходимый для хранения экземпляра структуры данных,*
- *форматы и интерпретация внутреннего представления*

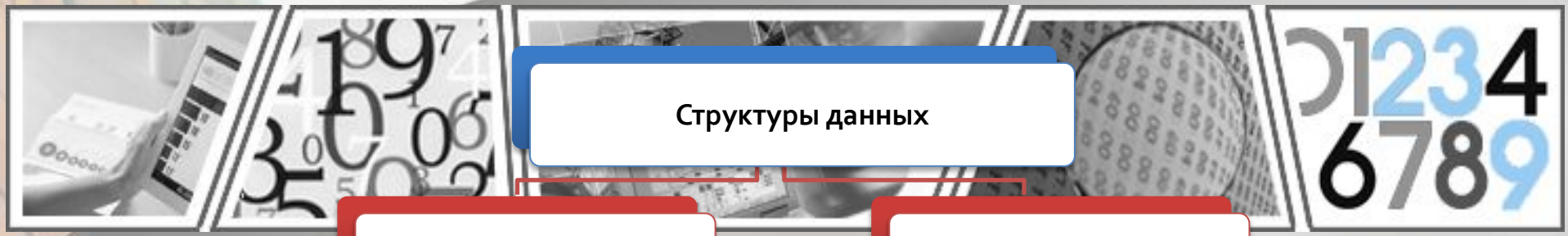
Физическая структура данных - способ физического представления данных в памяти машины и называется еще структурой хранения, внутренней структурой или структурой памяти

Классификация структур данных



Классификация типов данных по характер упорядоченности

Классификация структур данных



Классификация базовых и дополнительных структур данных

Классификация структур данных

- Структуры данных
  - Внутренние (в оперативной памяти)
    - Элементарные
      - Булевый
      - Числовой
      - Символьный
      - Указатель
      - .....
    - Составные
      - Линейные
        - Массив
        - Запись
        - Множество
        - Таблица
        - Линейный список
        - Стек
        - Очередь
        - Дек
        - .....
      - Нелинейные
        - Слоеный список
        - Мульти-список
        - Дерево
        - Граф
  - Внешние (на внешних устройствах)
    - Файл
    - База данных
    - .....

Классификация структур данных в зависимости от размещения физических структур и доступа к ним

Концепция типа данных

- *отображает особенности представления в компьютере данных различной природы*

Информация по каждому типу однозначно определяет:

- *структуру хранения данных указанного типа, то есть выделение памяти, представление данных в ней и метод доступа к данным;*
- *множество допустимых значений, которые может иметь тот или иной объект описываемого типа;*
- *набор допустимых операций, которые применимы к объекту описываемого типа*

Концепция типа данных

- *отображает особенности представления в компьютере данных различной природы*

Абстрактный тип данных (АТД) –

это формализованное описание (модель), определяющее организацию и набор возможных операций с описываемыми данными

Компьютерные данные - дискретные сообщения, *понятные компьютеру*

Для процессора компьютера любые данные представляют собой неструктурированную последовательность битов (поток битов)

Конкретная интерпретация этой последовательности зависит:

- *от программы*
- *от формы представления и структуры данных, которые выбраны программистом*
- *от решаемой задачи*
- *от удобства выполнения действий над данными*

Данные в программах

- *Непосредственные значения* - это неизменные объекты программы, которые представляют сами себя: числа (25, 1.34E-20), символы (‘A’, ‘!’), строки (‘Введите элементы матрицы’)
- *Константы* – это имена, закрепляемые за некоторыми значениями (const pi=3.1415926)
- *Переменные* - это объекты, которые могут принимать значение, сохранять его без изменения, и изменять его при выполнении определенных действий (var k:integer, x:real, a:array[1..3,1..5]);
- *Значения выражений и функций* – это записанные определённым способом правила вычисления значений: k*x+ sqrt (x)

Концепция типа данных

- *отображает особенности представления в компьютере данных различной природы*

Наиболее часто используют предопределенные скалярные типы:

- *целый (integer)*
- *вещественный (real)*
- *символьный (char)*
- *логический (boolean)*

Концепция типа данных

- *Тип* **integer**
  - *Целочисленные точные значения*
  - *Примеры: 73, -98, 5, 19674*
  - *Машинное представление: формат с фиксированной точкой*
  - *Диапазон значений определяется длиной поля*
  - *Операции: +, -, *, div, mod,=, <, и т.д.*

Концепция типа данных

- *Тип* **real**
  - *Нецелые приближенные значения*
  - *Примеры: 0.195, -91.84, 5.0*
  - *Машинное представление: формат с плавающей точкой*
  - *Диапазон и точность значений определяется длиной поля*
  - *Операции: +, -, *, /, =, <, и т.д.*

Концепция типа данных

- *Тип* **char**
  - *Одиночные символы текстов*
  - *Примеры: ‘a’, ‘!’, ‘5’*
  - *Машинное представление: формат ASCII*
  - *Множество значений определяется кодовой таблицей и возможностями клавиатуры*
  - *Операции: +, =, <, и т.д.*

Концепция типа данных

- *Тип* **boolean**
  - *Два логических значения false и true. Причем, false<true*
  - *Машинное представление — нулевое и единичное значение бита: false кодируется 0, true — 1*
  - *Операции: ¬, ∨, ∧, =, < и т.д.*

***Типы данных**, встроенные в язык программирования высокого уровня, являются базой для конструирования производных структур*

Поэтому для понимания организации производных структур на логическом и физическом уровнях важно иметь четкое представление об организации базовых типов данных

Основы организации данных на физическом уровне

- *Оперативная память* - *носитель информации, обрабатываемой в компьютере*
- *Бит* - *минимальная единица информации в оперативной памяти*
- *Бит может быть выключен, так что его значение есть нуль, или включен, тогда его значение равно единице*
- *Байт* - *группа (последовательность) из восьми битов*

Основы организации данных на физическом уровне

- *Доступ* к хранимой информации осуществляется побайтно
- *Оперативная память* - конечная последовательность байт
- *Физический адрес ячейки памяти (или просто адрес)* - номер байта в этой последовательности

  Физический адрес используется для получения доступа к конкретной ячейке

- *Байт* - минимально адресуемая ячейка памяти
- *Байт, слово, двойное слово* - типы ячеек памяти

Типы ячеек памяти

- ***Байт*** ***- восемь последовательно расположенных битов, пронумерованных от 7 до 0, при этом бит 0 является самым младшим значащим битом***

Количество бит составляющих ячейку памяти называется ее разрядностью

Типы ячеек памяти

- *Слово* - *последовательность из двух байт, имеющих последовательные адреса*
  - *Размер слова – 16 бит; биты в слове нумеруются от 15 до 0*
  - *Байт, содержащий нулевой бит, называется младшим байтом, а байт, содержащий 15-й бит, – старшим байтом*
  - *Адресом слова считается адрес его младшего байта*
  - *Адрес старшего байта может быть использован для доступа к старшей половине слова*

Типы ячеек памяти

- *Двойное слово* - последовательность из четырех байт (32 бита), расположенных по последовательным адресам
  - Нумерация бит производится от 31 до 0
  - Слово, содержащее нулевой бит, называется *младшим* словом, а слово, содержащее 31-й бит, *старшим* словом
  - Младшее слово хранится по меньшему адресу
  - Адресом двойного слова считается адрес его младшего слова
  - Адрес старшего слова может быть использован для доступа к старшей половине двойного слова

Логическая интерпретация типов данных

С точки зрения логической интерпретации выделяют следующие типы данных на физическом уровне:

- *целые числа без знака*
- *целые числа со знаком*
- *вещественные числа различной точност*
- *указатель*
- *цепочка байт*
- *символ*
- *строка*

Логическая интерпретация типов данных

- *Целые числа без знака* - *двоичное значение без знака, размером 8, 16 или 32 бита*
- *Основой представления является запись целого без знакового числа в двоичной системе счисления, где каждый бит соответствует двоичной цифре*
- *Числовой диапазон для этого типа следующий:*

• байт	от 0 до $2^8 - 1 = 255$;
• слово	от 0 до $2^{16} - 1 = 65\,535$;
• двойное слово	от 0 до $2^{32} - 1 = 4\,294\,967\,295$.

Логическая интерпретация типов данных

- *Целые числа со знаком* - *двоичное значение со знаком, размером 8, 16 или 32 бита*
- *Знак в этом двоичном числе содержится в 7, 15 или 31-м бите соответственно*
- *Ноль в этих битах в операндах соответствует положительному числу, а единица – отрицательному*
- *Целые числа со знаком представляются в дополнительном коде*

Логическая интерпретация типов данных

- *Дополнительный код числа* -

  фиксированное количество разрядов (n), образуется путем сложения соответствующего числа с числом 2^n с последующим взятием остатка от деления на 2^n

- ***В этом случае результат операции называется дополнением исходного числа до 2^n***

Логическая интерпретация типов данных

- *Дополнительный код числа*

Рассмотрим пример представления целых чисел со знаком в дополнительном коде

Пусть разрядность чисел составляет 8 бит. Число до которого будет строится дополнение есть 2^8 =256. Положительное число 3 в дополнительном коде есть

$$(256+3) \bmod 256 = 3 = 00000011_2.$$

Отрицательное число –3 в дополнительном коде есть

$$(256-3) \bmod 256 = 253 = 11111101_2.$$

Здесь и далее символ mod есть операция взятия остатка от деления

Логическая интерпретация типов данных

- *Правило представлений чисел в дополнительном коде*

1. *если число положительное, то его двоичная запись из n разрядов совпадает с дополнительным кодом;*
2. *если число отрицательное, то*
   - *необходимо записать модуль этого числа в двоичной системе счисления с использованием n разрядов*
   - *каждый бит этой записи проинвентировать (заменить нулевые биты на единичные и наоборот)*
   - *к полученному результату прибавить единицу*

Логическая интерпретация типов данных

- *Правило представлений чисел в дополнительном коде*

  Модуль числа –3 в двоичном виде из 8 цифр есть 00000011_2.

  После инвертирования получим 11111100_2.

  После прибавления единицы : 11111101_2, что соответствует представлению числа –3 в дополнительном коде (см. предыдущий пример)

Логическая интерпретация типов данных

- *Целые числа со знаком*

- *Числовые диапазоны для представления целых чисел со знаком:*

• 8-разрядное целое	от – 128 до 127;
• 16-разрядное целое	от – 32 768 до 32 767;
• 32-разрядное целое	от -2^{31} до $2^{31}-1$.

Логическая интерпретация типов данных

- *Вещественные числа –*
  - *на физическом уровне представляются в формате с плавающей точкой*
  - *кодирование чисел в виде трех составляющих: знака, мантиссы и порядка*
  - *В соответствии со стандартом IEEE 754 (Standard Floating Point Representations) существуют три формата: 32-битный, 64-битный и 80-битнй форматы представления чисел с плавающей точкой*

Логическая интерпретация типов данных

- *Вещественные числа –*
  - *32-битный формат называется форматом с одинарной точностью*
  - *Вещественные числа в этом формате занимают 4 байта, из которых старший бит кодирует знак числа, следующие за ним 8 бит – порядок числа и, наконец оставшиеся 23 бита – мантиссу*

1	8	23
s	*e*	*m*
знак	порядок	мантисса

Логическая интерпретация типов данных

- *Вещественные числа –*
  - *32-битный формат называется форматом с одинарной точностью*
  - *Формула для вычисления значения числа для этого представления*

$$v = \begin{cases} (-1)^s \cdot 2^{e-127} \cdot 1.m, & \text{если } 0 < e < 255; \\ (-1)^s \cdot 2^{126} \cdot 0.m, & \text{если } e = 0 \text{ и } m \neq 0; \\ (-1)^s \cdot 0, & \text{если } e = 0 \text{ и } m = 0; \\ (-1)^s \cdot Inf, & \text{если } e = 255 \text{ и } m = 0; \\ NaN, & \text{если } e = 255 \text{ и } m \neq 0. \end{cases}$$

Символ Inf - бесконечность
Символ NaN – нечисловое значение

Диапазон от $1.5 \cdot 10^{-45}$ до $3.4 \cdot 10^{+38}$

Логическая интерпретация типов данных

- *Вещественные числа –*
  - *64-битный формат называется форматом с двойной точностью*
  - *Вещественные числа в этом формате занимаю 8 байт, из которых старший бит кодирует знак числа, следующие за ним 11 бит – порядок числа и, наконец оставшиеся 52 бита – мантиссу*

Логическая интерпретация типов данных

- *Вещественные числа –*
  - *64-битный формат называется форматом с двойной точностью*
  - *Формула для вычисления значения числа для этого представления*

$$v = \begin{cases} (-1)^s \cdot 2^{e-1023} \cdot 1.m, & \text{если } 0 < e < 2047; \\ (-1)^s \cdot 2^{1022} \cdot 0.m, & \text{если } e = 0 \text{ и } m \neq 0; \\ (-1)^s \cdot 0, & \text{если } e = 0 \text{ и } m = 0; \\ (-1)^s \cdot Inf, & \text{если } e = 2047 \text{ и } m = 0; \\ NaN, & \text{если } e = 2047 \text{ и } m \neq 0. \end{cases}$$

Символ Inf - бесконечность
Символ NaN – нечисловое значение

Диапазон от $5.0 \cdot 10^{-324}$ до $1.7 \cdot 10^{+308}$

Логическая интерпретация типов данных

- *Вещественные числа –*
  - *80-битный формат называется расширенным форматом с двойной точностью*
  - *Вещественные числа в этом формате занимаю 10 байт, из которых старший бит кодирует знак числа, следующие за ним 15 бит – порядок числа и, наконец оставшиеся 63 бита – мантиссу*

Логическая интерпретация типов данных

- *Вещественные числа –*
  - *80-битный формат называется расширенным форматом с двойной точностью*
  - *Формула для вычисления значения числа для этого представления*

$$v = \begin{cases} (-1)^s \cdot 2^{e-16383} \cdot 1.m, & \text{если } 0 < e < 32767; \\ (-1)^s \cdot 2^{16382} \cdot 0.m, & \text{если } e = 0 \text{ и } m \neq 0; \\ (-1)^s \cdot 0, & \text{если } e = 0 \text{ и } m = 0; \\ (-1)^s \cdot Inf, & \text{если } e = 32767 \text{ и } m = 0; \\ NaN, & \text{если } e = 32767 \text{ и } m \neq 0. \end{cases}$$

Символ Inf - бесконечность
Символ NaN – нечисловое значение

Диапазон от $3.4 \cdot 10^{-4932}$ до $1.1 \cdot 10^{+4932}$

Логическая интерпретация типов данных

- *Указатель –*
  - *целочисленное значение, содержащее адрес в оперативной памяти*
- *Цепочка –*
  - *некоторый непрерывный набор байтов, или слов максимальной длиной до 64 Кбайт*
- *Символ –*
  - *байт, в который записывается код символа – целое от 0 до 255. В ЭВМ используется система кодировки ASCII (American Standard Code for Information Interchange)*

Классификация базовых типов и структур данных

Несмотря на многолетнее использование типов данных в отечественном программировании, так и не сложилась устойчивая и общепринятая русскоязычная терминология

КАТЕГОРИИ ТИПОВ ДАННЫХ:

- встроенные типы данных
- уточняемый тип данных
- перечисляемые типы данных
- конструируемые типы (составные)
- указательные типы
- определяемый пользователем тип данных

Классификация базовых типов и структур данных

- **встроенные типы данных** - типы, предопределенные в языке программирования, операции над значениями которых напрямую поддерживаются командами компьютеров

В современных компьютерах к таким "машинным" типам относятся:

- *целые числа разного размера (от одного до восьми байт)*
- *булевские значения (поддерживаемые обычно за счет наличия признаков условной передачи управления)*
- *числа с плавающей точкой одинарной и двойной точности (обычно четыре и восемь байт соответственно)*

Классификация базовых типов и структур данных

- **встроенные типы данных**

Тип CHARACTER (или CHAR) в разных языках – это:

1) *набор печатных символов из алфавита, зафиксированного в описании языка (для большинства языков англоязычного происхождения этот алфавит соответствует кодовому набору ASCII) ;*

Языки линии Паскаль

- *для значений типа CHAR определены только операции сравнения в соответствии с принятым алфавитом.*

Например, при использовании ASCII выполняются соотношения

'0' < '1' < …< '9' < 'A' < 'B' < …< 'Z' < 'a' < 'b' < …< 'z';

если '0' < x < '9', то значение x – цифра;

если 'A' < x < 'Z', то значение x – прописная буква;

если 'a' < x < 'z', то значение x – строчная буква и т.д.

Арифметические операции над символьными значениями не допускаются

Классификация базовых типов и структур данных

- **встроенные типы данных**

Тип CHARACTER (или CHAR) в разных языках – это:

2) произвольная комбинация нулей и единиц, размещаемых в одном байте

Языки линии Си

- *константами типа CHAR по-прежнему могут быть печатные символы из принятого в языке алфавита, но возможно использование и числовых констант, задающих желаемое содержимое байта*

В этом случае, как правило, над значениями типа CHAR возможно выполнение не только операций сравнения, но и операций целочисленной арифметики

В современных компьютерах, как правило, поддерживается целочисленная байтовая арифметика, обеспечивающая как первую, так и вторую интерпретацию типа CHAR

Классификация базовых типов и структур данных

- **встроенные типы данных**

Тип BOOLEAN (или BOOL)

Содержит два значения – TRUE (истина) и FALSE (ложь)

- *Для всех типов данных, для которых определены операции сравнения, определены также и правила, по которым эти операции сравнения вырабатывают булевские значения*

Над булевскими значениями возможны операции :

- *конъюнкции (&& или AND)*
- *дизъюнкции (|| или OR)*
- *отрицания (~ или NOT)*

Классификация базовых типов и структур данных

- **встроенные типы данных**

Тип BOOLEAN (или BOOL)

Над булевскими значениями возможны операции :

- *конъюнкции (&& или AND)*
- *дизъюнкции (|| или OR)*
- *отрицания (~ или NOT)*

TRUE AND TRUE =TRUE; TRUE AND FALSE = FALSE; FALSE AND TRUE = FALSE; FALSE AND FALSE = FALSE;	TRUE OR TRUE = TRUE; TRUE OR FALSE = TRUE; FALSE OR TRUE = TRUE; FALSE OR FALSE = FALSE;	NOT FALSE = TRUE; NOT TRUE = FALSE.

Таблицы истинности

Классификация базовых типов и структур данных

- **встроенные типы данных**

Тип BOOLEAN (или BOOL)

В языках линии Си прямая поддержка булевского типа данных отсутствует, но имеется логическая интерпретация значений целых типов

Значением операции сравнения может быть "0" (FALSE) или "1" (TRUE)

Значение целого типа "0" интерпретируется как FALSE, а значения, отличные от нуля, как TRUE

В остальном все работает как в случае наличия явной поддержки булевского типа

Классификация базовых типов и структур данных

- **встроенные типы данных**

Тип целых чисел

С помощью целых чисел может быть представлено количество объектов, являющихся дискретными по своей природе

При определении типа целых чисел обычно стремятся к тому, чтобы множество его значений было симметрично относительно нуля

Диапазон значений	Машинное представление
–128…127	8 бит со знаком
–32768…32767	16 бит со знаком
–2147483648…2147483647	32 бита со знаком
0...255	8 бит без знака
0…65535	16 бит без знака
$-2^{63}+1...2^{63}-1$	64 бита со знаком

Диапазон возможных значений целых типов зависит от их внутреннего представления, которое может занимать 1, 2 или 4 байта

Классификация базовых типов и структур данных

- **встроенные типы данных**

Тип чисел с плавающей точкой

Базовое название REAL или FLOAT

Значение вещественных типов определяет число лишь с некоторой конечной точностью, зависящей от внутреннего формата вещественного числа

Диапазон значений	Значащие цифры	Размер в байтах
$2{,}9\cdot10^{-39}...1{,}7\cdot10^{38}$	11–12	6
$1{,}4\cdot10^{-45}...3{,}4\cdot10^{38}$	7–8	4
$4{,}9\cdot10^{-324}...1{,}8\cdot10^{308}$	15–16	8
$3{,}1\cdot10^{-4944}...1{,}2\cdot10^{4932}$	19–20	10

Классификация базовых типов и структур данных

- **встроенные типы данных**

Операции над данными числовых типов

Четыре основных операции:

- *создание,*
- *уничтожение,*
- *выбор,*
- *обновление*

Арифметические операции:

- *сложение,*
- *вычитание,*
- *умножение,*
- *деление*

Операции сравнения: $>, <, \geq, \leq, =$

Поскольку вещественные числа представляются в памяти с некоторой (не абсолютной) точностью, сравнения их не всегда могут быть абсолютно достоверны

Структура типов данных в языке Паскаль