Машинное представление данных

Хранение информации

В ЭВМ обычно выделяют три основных вида запоминающих устройств:

сверхоперативная, оперативная и внешняя память.

Обычно сверхоперативная память строится на регистрах (в частности, регистрах процессора). Регистры используются для временного хранения и преобразования информации.

Оперативная память предназначена для запоминания более постоянной по своей природе информации, но после выключения ЭВМ она исчезает. Важнейшим свойством оперативной памяти является адресуемость. Это означает, что каждая ячейка памяти имеет свой идентификатор, однозначно идентифицирующий ее в общем массиве ячеек памяти. Этот идентификатор называется адресом. Адреса ячеек являются операндами тех машинных команд, которые обращаются к оперативной памяти. В подавляющем большинстве современных вычислительных систем единицей адресации является байт - ячейка, состоящая из 8 двоичных разрядов. Определенная ячейка оперативной памяти или множество ячеек могут быть связаны с конкретной переменной в программе. Для выполнения арифметических вычислений, в которых участвует переменная, необходимо, чтобы до начала вычислений значение переменной было перенесено из ячейки памяти в регистр. Если результат вычисления должен быть присвоен переменной, то результирующая величина снова должна быть перенесена из соответствующего регистра в связанную с этой переменной ячейку оперативной памяти.

Внешняя память служит прежде всего для долговременного хранения данных. Характерным для данных на внешней памяти является то, что они могут сохраняться там даже после завершения создавшей их программы и могут быть впоследствии многократно использованы той же программой при повторных ее запусках или другими программами. Внешняя память используется также для хранения самих программ, когда они не выполняются.

При программировании необходимо понимать машинное представление данных, т. е. знать как информация физически размещается в памяти вычислительной машины (в первую очередь, в оперативной).

Машинное представление данных

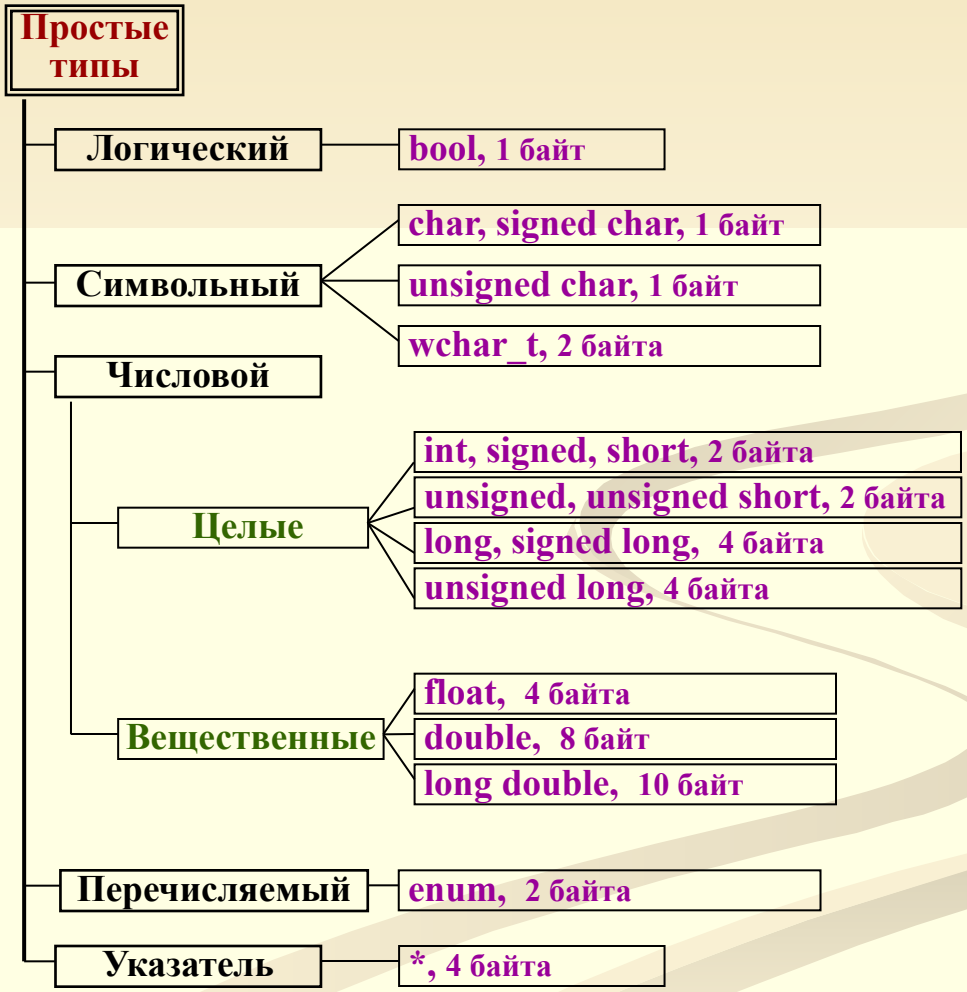
Различные структуры данных имеют разное машинное представление, особенно базовые (простые) структуры, из которых строятся более сложные конструкции данных.

Машинное представление данных

C / C++

Типовые размеры памяти для простых типов данных

В языках программирования простые структуры описываются простыми (базовыми) типами. Размер памяти, необходимый для данных того или иного типа, может быть разным не только в разных языках программирования, но и в разных реализациях одного и того же языка.



Адресация данных

Для IBM совместимых ПЭВМ и MS DOS

Оперативная память (ОП) ПЭВМ состоит из ячеек размером в 1 байт, каждая ячейка имеет собственный уникальный номер – адрес. Память разделяется на сегменты размером 2^{16} = 65 536 байт или 64 Кб, что определяется архитектурой микропроцессоров 80x86 (16-разрядных). [32-х и 64-х-разрядная адресация поддерживается в операционных системах Windows и UNIX / Linux.]

Адрес каждого байта ОП формируется из двух слов (каждое размером по 16 бит – по 2 байта) – адреса сегмента и смещения (смещение указывает на сколько байт от начала сегмента сдвинут адрес конкретного байта).

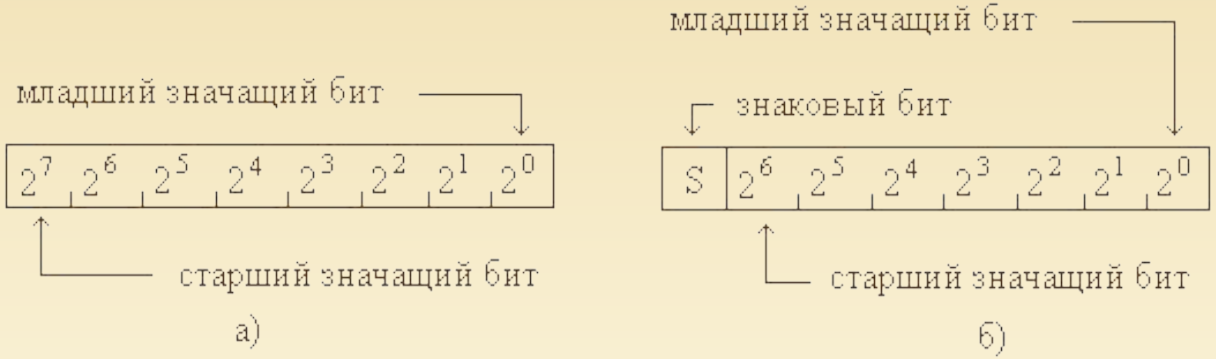
Адрес байта формируется следующим образом: адрес сегмента смещается на 4-е двоичных разряда (бита) влево и к нему прибавляется смещение, таком образом физический адрес байта составляет 20 бит и с его помощью можно адресовать 2^{20} байт (1 Мбайт):

Указатель занимает в памяти 4 байта (32 бита) и адресует первый байт данных, например, первый байт первого элемента массива.

Указатель занимает в памяти 4 байта (32 бита) хотя собственная длина указателя – 20 бит и он помещается в 3 байта, НО обычно память выделяется словами по 2 байта (16 бит) и выравнивается на границу слова – т.е под указатель выделяется 2 слова (а не 1,5).

Машинное представление данных

Машинное представление байта



Байт

а - беззнаковый байт, б - знаковый байт

В байте со знаком, для представления числа можно использовать не 8, а только 7 бит (разрядов). Восьмой же бит является *знаковым (S – sign)*, т.е. индицирует знак хранимого в байте числа; 1 в этом разряде, обычно, соответствует отрицательному числу, а 0 – положительному.

Операции с таким представлением наглядны, но не экономичны. Для оперирования со знаком числа требуется выполнять специальные алгоритмы операции, а это затраты машинного времени.

Более экономным является представление отрицательных чисел в дополнительном коде, который формируется следующим образом:

- модуль отрицательного числа записать в прямом коде, в неиспользуемые старшие биты записать нули;
- сформировать обратный код числа, для этого нуль заменить единицей, а единицу заменить нулем;
- к обратному коду числа прибавить единицу;
- знаковый (старший) разряд числа АВТОМАТИЧЕСКИ станет равным 1.

Пример: для числа -33 в формате integer:

0000000000100001 - прямой код
1111111111011110 - обратный код
\+ _____1
1111111111011111 - дополнительный код

Положительные числа хранятся в прямом коде, при этом знаковый (старший) разряд равен 0. Аналогично представляются целые числа других типов.

Машинное представление данных

Дополнительный код позволяет свести операцию вычитания к сложению положительного числа в прямом коде и отрицательного в обратном.

Пример: **Сложим +1 и -1:**

00000001 - в прямом двоичном коде
11111111 - в дополнительном коде
00000000 - перенос разряда дает в результате +1-1=0

(левый разряд переполнения – отбрасывается)

Тот же принцип дополнительного кода для отрицательных чисел можно использовать и в компьютерном представлении шестнадцатеричных (десятичных) чисел: для каждого разряда цифра X заменяется на 15-X (9-X), и к получившемуся числу добавляется 1.

Примеры:

- **Десятичные числа – при использовании четырёхзначных чисел -0081 заменяется на 9919 (9919+0081=0000, пятый разряд отбрасывается),**
- **16-ричные числа в дополнительном коде – 7F3C - прямой код**

  80C3 - обратный код
  +1
  80C4 – дополнительный код

Для отображения шестнадцатеричных чисел (в частности, в редакторе Total Commander и в других редакторах, например, FAR) используются восемь двоичных разрядов байта разбитые на две группы по четыре бита. Четырьмя разрядами двоичной системы счисления представляется одна цифра шестнадцатеричной системы счисления.

В шестнадцатеричной системе счисления число представляется в виде суммы степеней числа 16. Для изображения числа используется шестнадцать цифр: десять обычных десятичных цифр {0,1,2,3,4,5,6,7,8,9} и шесть латинских заглавных букв {A,B,C,D,E,F}.

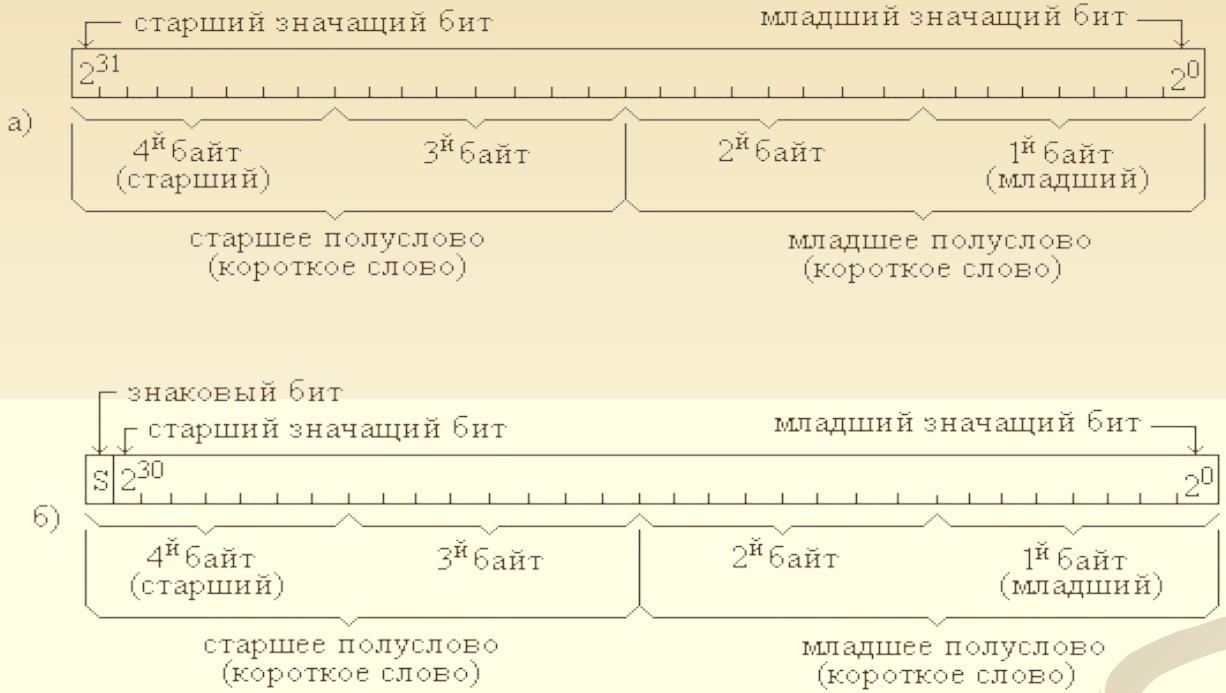
Таким образом содержимое байта отображают двузначным числом в шестнадцатеричной системе счисления (младший разряд – справа).

Например, $0110\ 1101_2 = 6D_{16}$

$1011\ 0010_2 = A2_{16}$

Машинное представление данных

Машинное представление совокупности байт



Двойное слово 32 (разряда)

а - беззнаковое, б - знаковое

В 16-разрядных процессорах слово состоит из 2-х байт. В 32- разрядных процессорах слово состоит из 4-х байт. Двойное слово, как и следует из названия, содержит ровно в два раза больше байт, чем просто слово. Как же называют набор из двух байт для 32-разрядного процессора? На больших ЭВМ - это полуслово. А на процессоре Intel 80x86 (младшие модели этих процессоров были 16-разрядными, а начиная с 80386 стали 32-разрядными) фирма Intel сохранила терминологию 16-разрядных моделей. В официальной документации на процессоры Intel словом, или коротким словом, называется набор из 2 байт, то есть *16 разрядов*. Это верно даже для Pentium. Набор из 4 байт, или *32 разряда*, называется двойным словом, или длинным словом. *Это сделано для единства терминологии, независимо от конкретной модели ПЭВМ.*

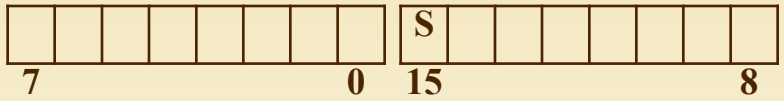
В процессорах Intel слова хранятся в памяти начиная с младшего байта, и за адрес слова принимается адрес младшего байта. То есть короткое слово 53C6 в памяти хранится так: C6, 53. А длинное слово 14AFB820 так: 20, B8, AF, 14. Т.е. младший байт числа – слева, а младший разряд в байте – справа.

Машинное представление данных

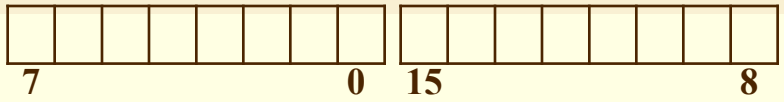
Целые числовые типы данных

C / C++

int, signed [int], short [int] – переменная хранится как слово (2 байта) со знаком



unsigned [int], unsigned short [int] – переменная хранится как слово (2 байта) без знака



long [int], signed long [int] – переменная хранится как двойное слово (4 байта) со знаком

unsigned long [int] – переменная хранится как двойное слово (4 байта) без знака

Машинное представление данных

C / C++

Логические типы данных

bool – переменная хранится как байт без знака

Символьные типы данных

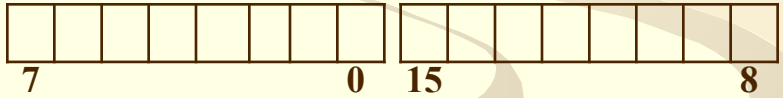
char, signed char – переменная хранится как байт со знаком

unsigned char – переменная хранится как байт без знака

wchar_t – переменная хранится как слово (2 байта) без знака

Перечисляемый тип данных

enum – переменная хранится как тип int (2 байта) без знака



Указатели

Тип указателя – это адрес ячейки памяти. В среде MS DOS адрес имеет размер 20 двоичных разрядов (бит).

А хранятся эти 20 бит в памяти в 4-х байтах, т.е. 32 бита: для ускорения вычисления адресов оперативной памяти выровнен на границу слова, слово - 2 байта и кратно степени двойки.

Машинное представление данных

Формат машинного представления вещественных чисел

Система вещественных чисел, применяемая при ручных вычислениях, предполагается бесконечно непрерывной. Это означает, что не существует никаких ограничений на диапазон используемых чисел и точность их представления. Для любого вещественного числа имеется бесконечно много чисел, которые больше или меньше его, а между любыми двумя вещественными числами также находится бесконечно много вещественных чисел.

Реализовать такую систему в технических устройствах невозможно. Во всех компьютерах размеры ячеек памяти фиксированы, что ограничивает систему представимых чисел. Ограничения касаются как диапазона, так и точности представления чисел, т.е. система машинных чисел оказывается конечной и и дискретной, образуя подмножество системы вещественных чисел.

Отсюда, в отличии от порядковых типов (все целые, символьный, логический), значения которых всегда сопоставляются с рядом целых чисел и, следовательно, представляются в памяти машины абсолютно точно, значение вещественных типов определяет число лишь с некоторой конечной точностью, зависящей от внутреннего формата вещественного числа.

Для вещественных чисел по стандарту ANSI/IEEE 754-1985 (IEEE Standard for Binary Floating-Point Arithmetic's) используется нормализованное представление со смещенным порядком. То есть число приводится к виду:

$$A = \pm M \cdot 2^{\pm P}, \quad ½ <= M < 1,$$

где M – мантисса, P – порядок. (Нормализованная запись отличного от нуля действительного числа – это запись вида $a = m*Q^p$, где p – целое число (положительное, отрицательное или ноль), а m – правильная Q-ричная дробь, у которой первая цифра после запятой не равна нулю, то есть $1/Q <= m <1$). Так как в результате старший разряд числа всегда равен единице, его обычно в памяти не хранят («скрытый бит» или «скрытая единица»). Порядок хранится в смещенном коде («модифицированный порядок» или «характеристика»): $P' = P+2^{n-1}$, где n – число разрядов машинного представления порядка. Это делается для того, чтобы характеристика всегда была положительной. Для представления нуля обнуляются все биты мантиссы и порядка.

Объясним ИНАЧЕ: Старший разряд двоичного представления вещественного числа всегда кодирует знак числа. Остальная часть разбивается на две части: мантиссу и экспоненту. Вещественное число имеет вид:

$$A = \pm S \cdot M \cdot 2^E,$$

где S - знаковый бит числа, E - экспонента, M - мантисса. Если $1/2 \le M < 1$, то такое число называется нормализованным. При хранении нормализованных чисел сопроцессор отбрасывает целую часть мантиссы (она всегда 1), сохраняя лишь дробную часть. Экспонента кодируется со сдвигом на половину разрядной сетки, таким образом, удается избежать вопроса о кодировании знака экспоненты. Т.е. при 8-битной разрядности экспоненты код 0 соответствует числу -127, 1 - числу -126, ..., 255 числу +126 (экспонента вычисляется как код 127).

Машинное представление данных

Формат машинного представления вещественных чисел

Стандарт IEEE-754 определяет три основных способа кодирования (типа) вещественных чисел:

Формат	Общая длина (байт / бит)	Знак мантиссы (бит)	Порядок (бит)	Мантисса (бит)	Смещение порядка	Диапазон представимых чисел	Точность в десятичной системе счисления	Особенность представления
Вещественное ординарной точности single precision	4 / 32	1	8	23	127_{10} $7FH_{16}$	$10^{-38}...10^{38}$	7 – 8 цифр	неявный бит F_0
Вещественное двойной точности double precision	8 / 64	1	10	53	1023_{10} $3FFH_{16}$	$10^{-308}...10^{308}$	15 – 16 цифр	неявный бит F_0
Вещественное расширенной точности extended precision	10 /80	1	15	64	16383_{10} $3FFFH_{16}$	$10^{-4932}...10^{4932}$	19 – 20 цифр	явный бит F_0

Пример: кодирование числа 178,625 в соответствии с IEEE-754.

$$178{,}625_{10} = 128 + 32 + 16 + 2 + 0{,}5 + 0{,}125 =$$
$$1\cdot2^7 + 0\cdot2^6 + 1\cdot2^5 + 1\cdot2^4 + 0\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 + + 1\cdot2^{-1} + 0\cdot2^{-2} + 1\cdot2^{-3} = 10110010{,}101_2$$

Его нужно нормализовать (привести в экспоненциальный вид):

$$1{,}78625E_{10}2 = 1{,}0110010101E_2111$$

В формате вещественного числа одинарной точности оно будет представлено так:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Знак | Смещенная экспонента: 111+01111111 | Дробная часть мантиссы 1,0110010101

Машинное представление данных

Вещественные числовые типы данных

C / C++

float

i – позиция неявной двоичной точки

double

i – позиция неявной двоичной точки

long double

1 –целочисленный бит мантиссы
(явный в long и неявный во float и double)

Алгоритм формирования машинного представления вещественного числа в памяти ЭВМ

1) Число представляется в двоичном коде.
2) Двоичное число нормализуется. При этом для чисел, больших единицы, плавающая точка переносится влево, определяя положительный порядок. Для чисел, меньших единицы, точка переносится вправо, определяя отрицательный порядок.
3) Затем с учетом типа вещественного числа определяется характеристика.
4) В отведенное поле памяти (в соответствии с типом числа) записываются мантисса, характеристика и знак числа. При этом необходимо отметить следующее:
(а) для чисел типа real (Pascal) характеристика хранится в младшем байте памяти, для чисел типа single, float, double, extended, long double - в старших байтах; (б) знак числа находится всегда в старшем бите старшего байта;
(в) мантисса всегда хранится в прямом коде; (г) целая часть мантиссы (*для нормализованного числа всегда равна 1*) для чисел типа single, real, float, double не хранится (является скрытой). В числах типа extended, long double все разряды мантиссы хранятся в памяти ЭВМ.

Машинное представление данных

Практическое занятие: C / C++

Анализ машинных форматов на примере файлов

```
Program DataType;
type
  StructInt = record
     ii : integer;      si : shortint;       li : longint;
     ch : char;
     by : byte;
          end;
var
 f : file of StructInt ;  (* типизированный файл *)
 St : StructInt;
begin
  assign (f, 'z:\datatype.dtp');    (* Создание файла *)
  rewrite (f);  (*Открыть файл в режиме перезаписи*)
  write ('Введите целое число  со знаком и
              нажмите Enter ->');      readln (St.ii);
  write ('Введите короткое  целое со знаком и
            нажмите Enter ->');      readln (st.si);
  write (' ведите длинное целое со знаком и
             нажмите Enter ->');      readln (st.li);
  write ('Введите символ (букву) и
               нажмите Enter ->');
    readln (st.ch);
   write (' Введите целое число без знака и
                 нажмите Enter ->');      readln (st.by);
write (f, st);  (*запись считанной структуры в файл*)
   close (f);     (* закрыть файл *)
   writeln ('Введенные числа и символ записаны в
файл ', 'z:\datatype.dtp');
   readln;
end.
```

Задание:

1. Вывести в файл datatype.dtc заданные преподавателем целые и символьные данные и просматривая их на экране в 16-ричном виде определить где какое число и символ (смотреть 16-ричное значение в редакторах FAR или Total Commander).
2. Запустить отладчик Borland C/C++, посмотреть в нём адреса вводимых чисел и определить сколько памяти отводится под каждое значение.

Машинное представление данных

Практическое занятие: C / C++

Анализ машинных форматов на примере файлов

```
Program DataType;
type
  StructInt = record
     ii : integer;      si : shortint;       li : longint;
     ch : char;
     by : byte;
          end;
var
 f : file of StructInt ;  (* типизированный файл *)
 St : StructInt;
begin
  assign (f, 'z:\datatype.dtp');    (* Создание файла *)
  rewrite (f);  (*Открыть файл в режиме перезаписи*)
  write ('Введите целое число  со знаком и
              нажмите Enter ->');      readln (St.ii);
  write ('Введите короткое  целое со знаком и
            нажмите Enter ->');      readln (st.si);
  write (' ведите длинное целое со знаком и
             нажмите Enter ->');      readln (st.li);
  write ('Введите символ (букву) и
               нажмите Enter ->');
    readln (st.ch);
   write (' Введите целое число без знака и
                 нажмите Enter ->');      readln (st.by);
write (f, st);  (*запись считанной структуры в файл*)
   close (f);     (* закрыть файл *)
   writeln ('Введенные числа и символ записаны в
файл ', 'z:\datatype.dtp');
   readln;
end.
```