Сложные структуры данных

Массивы. Строки переменной длины. Перечисления. Множества. Записи. Структуры. Объединения. Списки.

Массивы

- упорядоченные множества элементов одного типа, занимающих непрерывное пространство в памяти машины.

Упорядоченность элементов массива определяется набором целых чисел, называемых индексами, которые связываются с каждым элементом массива и однозначно определяют его положение среди остальных элементов этого массива.

$$X = \{x_{ij}: a \le i \le b \text{ и } c \le j \le d\}$$

$$\begin{pmatrix}
x_{a,c} & x_{a,c+1} & \dots & x_{a,d-1} & x_{a,d} \\
x_{a+1,c} & x_{a+1,c+1} & \dots & x_{a+1,d-1} & x_{a+1,d} \\
\dots & x_{ij} & \dots & & \\
x_{b-1,c} & x_{b-1,c+1} & \dots & x_{b-1,d-1} & x_{b-1,d} \\
x_{b,c} & x_{b,c+1} & \dots & x_{b,d-1} & x_{b,d}
\end{pmatrix}$$

Расположение в памяти

- По строкам: Быстый последний индекс

$$x_{a,c}\ x_{a,c+1} \cdots x_{a,d-1}\ x_{a,d}\ x_{a+1,c}\ x_{a+1,c+1} \cdots x_{a+1,d-1}\ x_{a+1,d} \cdots$$
$$x_{a+1,c}\ x_{a+1,c+1} \cdots x_{a+1,d-1}\ x_{a+1,d}\ x_{b,c}\ x_{b,c+1} \cdots x_{b,d-1}\ x_{b,d}$$

X+((i-a)*(d–c+1) + j-c)*Type X

- По столбцам:

$$x_{a,c}\ x_{a+1,c} \cdots x_{b-1,c}\ x_{b,c}\ x_{a,c+1}\ x_{a+1,c+1} \cdots x_{b-1,c+1}\ x_{b,c+1} \cdots$$
$$x_{a,d-1}\ x_{a+1,d-1} \cdots x_{b-1,d-1}\ x_{b,d-1}\ x_{a,d}\ x_{a+1,d} \cdots x_{b-1,d}\ x_{b,d}$$

Быстрый первый индекс

X+((j-c)*(b–a+1) + i-a)*Type X

Строки переменной длины

- Си

```
String  db  ‘Это строка С’,0
```

Произвольная длина

Длину необходимо вычислять

- Паскаль

```
String  db  Len-1,‘Строка Паскаля’
   Len = $ - String
```

Длина не более 255

Длина всегда известна без вычислений

Перечислимые типы данных

```
Azb enum a,b,c,d,e,f,g,h,i,j,k,l,n,m,o,p,q,r,s,t,v,u,w,x,y,z
        ; a=0, b=1, … , z=25

Azb enum a=61h,b,c,…
        ; a=61h, b=62h, …
        mov  al,azb     ; al=31
```

Число бит, необходимых для представления множества, с перечисленным числом компонент

Множество

– совокупность элементов, обладающих общим для них характеристическим свойством.
– конечная упорядоченная совокупность элементов, т.е. каждому элементу поставлено в соответствие натуральное число, являющееся номером элемента множества.

Множество можно моделировать набором бит в количестве равном числу элементов множества, когда значение i-ого бита отвечает наличию или отсутствию i–ого элемента в множестве.

Для множества мощностью n требуется n/8+1 байт

Выделение места для размещения множества

```
  Setof  macro    name,pw,x
  rr=pw/8
  if pw mod 8 ne 0
   rr=rr+1
  endif
  if rr eq 1
name label byte
  elseif rr eq 2
name label word
  elseif rr le 4
name label dword
  elseif rr le 8
name label qword
  else
```

```
 .err "Не могу создать множество такого размера“
exitm
endif
kk=0
while kk lt rr
   shablon=0
   irp   i,<x>
     if i/8 eq kk
       shablon=shablon or (1 shl (i mod 8))
     endif
   endm
   db    shablon
   kk=kk+1
endm
endm
```

Сегмент данных

```
.data
Alphabet enum A,B,C,D,E,F,G,H,I,J,K,L,N,M,O,P,Q,R,S,T,V,U,W,X,Y,Z
    SETOF VOWELS,Alphabet,<a,e,i,j,o,u>
    SETOF CONSONANTS,Alphabet,<b,c,d,f,g,h,k,l,m,n,p,q,r,s,t,v,w,x,y,z>
```

Сегмент команд

```
.code
main proc
 mov  ax,@data
 mov  ds,ax
 inmn VOWELS,B
 jnc short no
      beepspkr
 no:.exit     0
main   endp
       end     main
```

Проверка присутствия элемента в множестве

```
Inmn   macro  name,k
   ;; Результат в fc
  ifndef <name>
  .err ‘Имя &name не определено‘
   exitm
  elseifndef   <k>
  .err ‘Имя &k не определено‘
   exitm
  endif
  if     k gt (type name)*8
   clc
   exitm
  endif
  push ax
  push bx
  mov ax,k
  ror    ax,3
  shr    ah,5     ;; ah=номер бита,
         ;; al=номер байта
  mov  bl,al
  xor    bh,bh
  mov  bl,byte ptr name[bx]
  shr    ax,8
  bts    bx,ax
  pop   bx
  pop   ax
  endm
```

Вспомогательный макрос

```
beepspkr  macro times:=<1>
    push ax
    push dx
    mov  ah,2
    mov  dl,7
    rept   times
      int 21h
    endm
    pop  dx
    pop  ax
    endm
```

Записи

- наборы групп битовых полей внутри байта, слова или двойного слова.

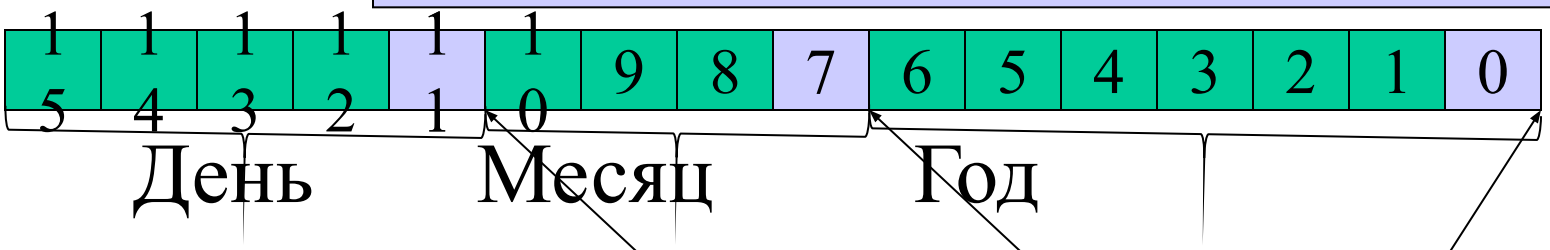
Формат описания шаблона:

Имя_шаблона RECORD *айтем[,айтем…]*

Имя:длина[=значение по ~~умолчанию~~]

Например:

Константа равная сдвигу поля от правого края



Date_format record *day*:5=1,*month*:4=1,*year*:7=0

Описание переменных типа запись

- Test
- Mi
- nth=4}

mov ax,test_date
and ax,mask month
shr ax,month

аписями

- Mask и

mov ax,mask year

getfild month bl,test_date

And test_date,not mask month
Or test_date,6 shl month

ot mask month

mov bl,6

; ax = 4

setfild month test_date,6

Структуры

- совокупности переменных различного типа.

Формат описания шаблона структуры:

Имя_шаблона struc
Описатель переменной
[…
Описатель переменной]
Имя_шаблона ends

Например:

```
Complex   str
 Redd 0.
 Imdd 0.
Complex   enc
```

конст
рас

```
Mov ax,type complex.re
    ; ax = 4
Mov ax,type complex
    ; ax = 8
```

Описание переменных формата структура

- Cnul complex <>
- Ced complex <1.,>
- Ci complex {im=1.0}
- Carray complex 10 dup <?,?>

Доступ к полям структуры

- Прямая адресация
  Mov eax,Cnul.Re
  mov Cnul.Im,eax
- Косвенная адресация
  Mov bx,offset Ci
  Mov eax,[bx].Im

Объединения

- наложение переменных различного типа.

Формат описания шаблона объединения:

Имя_шаблона Union
описатель переменной
[…
описатель переменной]
Имя_шаблона ends

- Правила работы те же, что и со структурами
- Другой способ – использование директивы

Имя LABEL *тип*

Списки

- совокупность элементов типа структура, расположенных в произвольных местах памяти, связанных друг с другом через поля связи – переменные в которых хранятся ссылки (адреса) на следующий элемент.
- Последний элемент списка (не связанный с другими) хранит в таком поле пустую ссылку, называемую nil.
- Ссылка на первый элемент списка хранится в переменной, которую называют указателем на вершину списка.

Примерный формат элемента списка (aitem).

- ```
  filds   struc
  fild1   dw?
    …
  filds   ends
  ```

- ```
  Aitem  struc
    list filds <>
    next dw ?
  Aitem  ends
  ```

Пример устройства «диспетчера» кучи

```
.model small
.386
nil = -1
Heap_size = 64*1024

filds  struc
fild1  db 8 dup(?)
fild2  db ?
filds  ends

aitem    struc
list     filds  <>
next     dw     ?
aitem    ends
```

Пример устройства «диспетчера» кучи

```
.data
Status   db Heap_size/8 dup(0)
top      dw    ?
str1     db 'String 1'
str2     db 'String 2'
str3     db 'String 3'

.stack   256

Heap segment    para
Htop   db   Heap_size dup(?)    ; Куча
Heap ends

.code
```

Создание нового элемента списка

```
new   macro    size
   ifndef  Heap
      .err
      exitm
   elseifndef     Status
      .err
      exitm
   endif
   mov ax,size ; Размер искомого свободного участка
   call  find
   endm
```

Удаление элемента списка

```
Delite  macro adr,size
   mov bx,adr  ; Адрес освобождаемой памяти в bx
   mov cx,size ; Размер освобождаемой области
   call  clear
   endm


Start   macro
   assume   es:Heap
   mov  ax,@data
   mov  ds,ax
   mov  ax,Heap
   mov  es,ax
   xor  ax,ax
   endm
```

Сохранение и загрузка регистров

```
SaveReg  macro
  r1,r2,r3,r4,r5,r6,r7,r8,r9
  ifnb    <r1>
    push  r1
    SaveReg
  r2,r3,r4,r5,r6,r7,r8,r9
  endif
  endm

LoadReg  macro
  r1,r2,r3,r4,r5,r6,r7,r8,r9
  irp
  k,<r9,r8,r7,r6,r5,r4,r3,r2,r1>
  ifnb   <k>
```

Основная программа

```
main  proc
.Start
   new    %type aitem    ; Найти подходящую область памяти для
                         ; размещения элемента списка
   mov    top,ax          ; Указатель на вершину списка
; Первый элемент
   mov    bx,ax
   mov    cx,8
   lea si,str1
   lea di,[bx].list.fild1
   rep movsb          ;заполнение поля fild1 первого элемента
   new    %type aitem
   mov    es:[bx].next,ax     ; адрес второго элемента
```

Создание 2 и 3 элементов

```
mov    bx,ax
mov    cx,8
lea si,str2
lea di,[bx].list.fild1
rep movsb          ;заполнение поля fild1 второго элемента
new    %type aitem
mov    es:[bx].next,ax      ; адрес третьего элемента
mov    es:[bx].list.fild2,'$'
mov    bx,ax
mov    cx,8
lea si,str3
lea di,[bx].list.fild1
rep movsb                   ;заполнение поля fild1 третьего
элемента
mov    es:[bx].next,nil; конец списка
mov    es:[bx].list.fild2,'$'
```

Удаление элемента и печать списка

```
mov    bx,top
delite es:[bx].next,%type aitem
mov    bx,top
mov    di,es:[bx].next
mov    ax,es:[di].next
mov    es:[bx].next,ax
mov    bx,top          ; Адрес текущего элемента списка
cont: cmp    bx,nil
jz   fin       ; достигнут конец списка
lea dx,[bx].list.fild1
push   ds
push   es
pop    ds
mov    ah,09h
int  21h    ;Печать тестового поля текущего элемента
pop    ds
mov    bx,es:[bx].next
jmp    cont
fin: .exit   0
main   endp
```

Процедура поиска места в куче

```
   find    proc  ; ax - размер требующейся памяти в байтах
   SaveReg bx,cx,dx
   mov   cx,Heap_size/8   ; кол-во байт под статус
   mov   bx,0
   push  ax      ; сохранить объем памяти в стеке
m0:  cmp  status[bx],0ffh ; проверка на все единицы
   jz   next1
   mov   dl,1            ; 1 в левый бит
m7:  test   status[bx],dl    ; проверка бита
   jz   m1
   pop   ax          ; текущий бит равен 1 - заново
   push  ax
   jmp   short m2
m1:  dec   ax          ; найден еще один нулевой бит
   jz   yes           ; найден нужный объем памяти
m2:  shldl,1          ; маска для следующего бита
   jz   m3            ; нужно перейти к новому байту
   jmp   m7
```

Процедура поиска места в куче

```
   next1: pop   ax ; восстановление размера
   push   ax
m3:   incbx ; увеличение номера байта
   loop   m0          ; цикл по всем байтам
   jz   no
yes:   shlbx,3   ; вычисление номера последнего бита
   pop    cx      ; считан размер
m4:   incbx ; добавить сдвиг внутри байта
   shr dl,1
   jnz m4
   sub    bx,cx      ; номер первого бита поля
   mov    ax,bx
   push   a
   shr bx,3        ; номер байт
   and    ax,07h         ; и еще сдвиг на несколько бит
   mov    ah,1
   push   cx
   mov    cl,al
   shl ah,cl      ; сдвиг маски в позицию первого бита
```

Заполнение массива Status

```
   pop   cx
m6: or  status[bx],ah     ; заполнение 1  маски
   отводимого поля
   dec   cx
   jz   m5
   shl ah,1
   jnz m6
   inc bx
   mov  ah,1
   jmp   m6
no:   mov ax,-1
m5:  LoadReg  bx,cx,dx,ax
   ret
find  endp
```

Процедура освобождения памяти

```
clear   proc
SaveRegax,dx
mov     ax,bx
shr bx,3        ; номер байта
and     ax,07h          ; и еще сдвиг на несколько бит
mov     ah,0feh
push    cx
mov     cl,al
rol  ah,cl      ; сдвиг маски в позицию первого бита
pop     cx
m16: and status[bx],ah ; заполнение 1 маски
отводимого поля
dec     cx
jz   m15
rol  ah,1
cmp     ah,0feh
jnz m16
inc bx
jmp     m16
m15: LoadReg ax,dx
ret
clear endp
end     main
```

Результат: