

.386 ; Тип процессора

.model flat, stdcall ; Модель памяти и стиль вызова подпрограмм

option casemap: none ; Чувствительность к регистру

; --- Подключение файлов с кодом, макросами, константами, прототипами функций и т.д.

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\msvcrt.inc

; --- Подключаемые библиотеки ---

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

; --- Сегмент данных ---

.data

i\_invite db "Input size of array = ", 0

i\_d db "Input array: ", 0

fmt db "%d", 0

msg db "Your array: ", 0

fmtout db "%d", 13, 10, 0

.code

; void enter\_arr(int \*arr, size\_t n);

enter\_arr proc

pushad

push offset i\_d

call crt\_printf

add esp, 4

mov eax, [esp+36] ; eax = arr

mov ecx, [esp+40] ; ecx = n

cmp ecx, 0

jle retl

loop1:

push eax ; сохраняем eax

push ecx

push eax

push offset fmt

call crt\_scanf ; вводим \*arr

add esp, 8

pop ecx

pop eax

add eax, 4 ; arr = arr + 4

loop loop1

retl:

popad

ret 8

enter\_arr endp

is\_pow\_3 proc

push ebx

mov eax, [esp+8] ; eax = x

cmp eax, 1 ; если x = 1, вернуть ложь

jle ret0

;степень тройки если eax and 11b = 0

and eax, 11b ; eax = eax AND ebx

cmp eax, 0

je ret0

jmp ret1

ret0:

mov eax, 0

jmp retlb

ret1:

mov eax, 1

retlb:

pop ebx

ret 4

is\_pow\_3 endp

; void del\_3(int \*adr, size\_t \*pn);

del\_3 proc

pushad

mov ebx, [esp+36] ; eax = adr

mov ecx, [esp+40] ; ecx = pn

push [ecx]

mov esi, 0 ; esi = i - индекс прохода по массиву

mov edi, 0 ; edi = j - индекс записи элементов

mov [ecx], esi

pop ecx

cmp ecx, 0

jle retl

loop1:

push dword ptr [ebx+esi\*4]

call is\_pow\_3

cmp eax, 0

jne nwrite

mov eax, [ebx+esi\*4]

mov [ebx+edi\*4],eax ; adr[j] = adr[i]

inc edi ; j++

mov edx, [esp+40]

mov [edx], edi ; \*pn = j

nwrite:

inc esi ; i++

loop loop1

retl:

popad

ret 8

del\_3 endp

out\_arr proc

pushad

mov eax, [esp+36]

mov ecx, [esp+40]

cmp ecx, 0

jle retl

loop1:

push eax

push ecx

push [eax]

push offset fmtout

call crt\_printf

add esp, 8

pop ecx

pop eax

add eax, 4

loop loop1

retl:

popad

ret 8

out\_arr endp

start:

mov eax, offset fmtout

add eax, 5

push eax

push offset i\_invite

call crt\_printf

add esp, 4

pop eax

push eax

push eax

push offset fmt

call crt\_scanf

add esp, 8

pop eax

push [eax]

add eax, 4

push eax

call enter\_arr

sub eax, 4

push eax

push offset msg

call crt\_printf

add esp, 4

pop eax

push eax

add eax, 4

push eax

call del\_3

sub eax, 4

push [eax]

add eax, 4

push eax

call out\_arr

add esp, 8

sub eax, 4

call crt\_\_getch

push 0

call ExitProcess

end start