Laborator 5

Structuri, aliniere

- Alinierea membrilor de date se realizeaza dependent de compilator;
- Datele membru sunt situate in memorie in ordinea in care s-a specificat in cod;
- In mod uzual, deplasamentul fiecarui membru fata de inceputul structurii este multiplu de dimensiunea sa;
- Structura se completeaza la sfarsit cu spatiu astfel incat marimea sa sa fie multiplu de cel mai mare dimensiune al vre-unui membru;

Exercitii rezolvate:

1. Se dau structurile **Str1** si **Str2**. Scrieti functiile care populeaza campurile acestora.

```
// Lab5.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <iostream>
using namespace std;

struct Str1 {
 char c;
 short s;
 int i;
};
/*
Str1 va fi aliniat astfel:
Primul byte (byte 0) va fi pentru c.
Bytii 2 si 3 vor fi pentru s. Byte-ul 1 ramane neutilizat.
Bytii 4,5,6,7 for fi pentru i.
*/
struct Str2 {
 char c;
 int i;
 short s;
};
/*
Str2 vi fi aliniat astfel:
Primul byte (0) va fi pentru c.
Bytii 1,2,3 vor ramane liberi.
  (pentru ca i are size 4 si trebuie sa inceapa de la o adresa multiplu de 4)
Bytii 4,5,6,7 for fi pentru i
Bytii 8 si 9 for fi pentru s.
*/
void generate(char c, short s, int i, Str1* x)
{
 _asm {
  //esi e adresa structurii.
  mov esi, [ebp+20]
  //in eax punem c. Folosim 4 bytes pentru ca stiva e aliniata.
  mov eax, [ebp+8]
  //Doar ultimul byte din cei 4 din eax contin c. Il punem in [esi+0]
  mov [esi], al

  //La fel, doar ca vom lua ultimii 2 bytes.
  mov eax, [ebp+12]
  mov [esi+2], ax
```

```
  //Aici ii punem pe toti 4. Nu conteaza ce returnam in eax.
  mov eax, [ebp+16]
  mov [esi+4], eax
 }
}


void generate_2(char c, short s, int i, Str2* x)
{
 _asm {
  /*Completati*/
  mov esi, [ebp+20]
  mov eax, [ebp+8]
  mov [esi], al
  mov eax, [ebp+12]
  mov [esi+8], ax
  mov eax, [ebp+16]
  mov [esi+4], eax
 }
}

int main()
{
 Str1 a;
 generate('a', 3, 5, &a);
 Str2 b;
 generate_2('b', 1, 10, &b);
 cout << "a: c=" << a.c << " s=" << a.s << " i=" << a.i << '\n';
 cout << "b: c=" << b.c << " s=" << b.s << " i=" << b.i << '\n';
    return 0;
}
```

Exercitii nerezolvate:

1. Completati functiile de mai jos. Ele reprezinta setteri si getteri pentru o structura.
Un getter si un setter a fost deja completat. Cititi comentariile de acolo. Nu aveti voie sa schimbati parametrii functiilor.
Implementati si o functie de copiere a unui elev. Functia ar trebui sa aiba 2 parametrii: un elev copiat, si o referinta catre un elev neinstantiat

```
#include "stdafx.h"
#include <iostream>
using namespace std;

struct Elev {
 char nume[20];
 int nota;
 short grupa;
 char semian;
};

void setNume(Elev *e, char* nume) {
 _asm {
  //Completati
 }
}

void setNota(Elev *e, int nota) {
 _asm {
  mov esi, [ebp+8]
  mov eax, [ebp+12]
  mov [esi+20], eax
 }
}

void setGrupa(Elev *e, short grupa) {
 _asm {
  //Completati
```

```
 }
}

void setSemian(Elev *e, char semian) {
 _asm {
  //Completati
 }
}

char* getNume(Elev *e) {
 _asm {
  //Completati
 }
}

int getNota(Elev e) {
 _asm {
  //Aici trebuie tinut cont ca se da push pe stiva la toata structura, nu la pointer!
  //Pe stiva se va incepe de la adresa [ebp+8]. Pana la adresa [ebp+27] inclusiv se a
  //De la adresa [ebp+28] se afla nota.
  mov eax, [ebp + 28]
 }
}

short getGrupa(Elev e) {
 _asm {
  //Completati
 }
}

char getSemian(Elev e) {
 _asm {
  //Completati
 }
}

int main() {
 Elev e = { "Georgel", 7, 3, 'A' };
 setNota(&e, 10);
 int nota = getNota(e);
 cout << "Nota este: " << nota << '\n';
 char* nume = getNume(&e);
 cout << "Numele este: " << (char*)nume << "\n";
}
```

2. Scrieti in C++ o structura care contine doua coordonate de tip int. Structura se va numi **Punct**
Scrieti in C++ o structura numita **Grafic** care contine un array de pointeri catre structuri de tip **Punct** si un **int**, lungimea acestui array.
Scrieti in asm o functie care primeste ca parametru un pointer catre o structura **Grafic**, si o referinta catre o structura **Punct**.
Dupa executarea functiei, parametrul de tip **Punct** va fi populat astfel incat coordonatele lui sa fie cele ale centroidului celorlalte puncte.
(Se vor face aproximatii astfel incat sa folositi doar **int**)
(x va fi media x-ilor si y media y-ilor)
Puteti folosi orice functii ajutatoare, atat timp cat sunt scrise tot in asm.

Varianta hardcore: **Punct** contine si un char*, care reprezinta label-ul acelui punct (numele).
Mai scrieti o functie care primeste ca parametru un pointer catre un **Grafic** si returneaza un pointer char* catre numele cel mai mare dpdv lexicografic al punctelor din grafic. Puteti apela functia strcmp, care respecta standarduul cdecl

Comments

You do not have permission to add comments.