Fundamentele programării
Curs 10

Șef lucr.dr.ing. GENGE Béla

Universitatea "Petru Maior", Departamentul de Informatică
Tîrgu Mureș, România
bela.genge@ing.upm.ro

Pointeri la pointeri

- Pointeri care adresează alți pointeri

Exemplu

```
int n = 10;
int* p1 = NULL;
int** p2 = &p1;
*p2 = &n;
**p2 = 0;
cout << &p2;
cout << p2;
cout << *p2;
cout << **p2;
cout << n;
```

Exerciții

- Să se aloce și dealoce dinamic memorie pentru o matrice de NxM întregi. Matricea este folosită pentru citiri de la tastatură.
- Să se citească N șiruri de caractere de la tastatură. Memoria va fi alocată dinamic printr-un pointer către un tablou de pointeri. Să se elibereze memoria ocupată în final.
- Să se elimine (dealoce memorie) pentru ultimul șir de caractere/ultima linie dintr-o matrice.

Funcții

- Noțiunea de funcție este fundamentală în limbajul C
- Execuția unui program poate fi organizată ca o ierarhie de apeluri de funcție
- Punctul de pornire: funcția `main()`

Elemente necesare utilizării funcțiilor

- Prototipul
- Definiția
- Apelul

Prototipul

- Asigură declararea unei funcții - necesar înainte de utilizare
- Declararea prototipului permite plasarea în cod a apelului unei funcții înainte de definirea acesteia
- De regulă prototipurile sunt incluse la începutul unui fișier sursă C, înainte de definirea oricărei funcții
- Prototipul ANSI-C include:

```
tip_returnat nume_funcție(listă_tipuri_param);
```

- Denumirea parametrilor este opțională

Prototipul - exemple

Exemple

```
void mesaj(void);
int suma(int, int);
int produsul(int a, int b);
int suma(int n, int v[100]);
```

Definiția

- Poate apărea o singură dată în program
- Definiția include: antetul și corpul funcției
- Sintaxa de definire ANSI-C:

```
tip_returnat nume_funcție(listă_param)
{declarații_variabile
instrucțiuni
}
```

- Atenție! Prototipul trebuie să coincidă cu antetul din definiție
- Lista de parametri formali - prototip și antet
- Lista de parametrii actuali - apel

Definiția - exemple

Exemplul1

```
void mesaj(void){
  printf("Limbajul C\n");
}
```

Exemplul2

```
int suma(int a, int b){
  return (a+b);
}
```

Exemplul3

```
int produsul(int a, int b){
  int c;
  c = a*b;
  return c;
}
```

Exemplul4

```
int suma(int n, int v[100]){
  int s = 0;
  for ( int i = 0 ; i < n ; ++i )
    s += v[i];
  return s;
}
```

Apelul

- Funcțiile sunt definite la adrese fixe de memorie
- Clasa de memorie implicită asociată este `extern`
- Rezultă că pot fi accesate atât din modulul curent cât și din alte module (e.g., modul = DLL)
- Pentru limitarea vizibilității doar la modulul curent se scrie la definiție înaintea tipului returnat `static`

Apelul

- Un apel este o expresie formată din numele funcției urmat de o pereche de paranteze între care se specifică lista parametrilor actuali
- Parametrii trebuie să corespundă ca număr și tip listei parametrilor formali
- Sintaxa:

```
nume_funcție(listă_param_actuali);
```

Apelul - exemple

Exemple

```
mesaj();
int s = suma(12,13);
int a = 10, b = 15;
printf("Produsul este: %d\n", produsul(a,b));
int n, v[100];
std::cout << suma(n, v);
```

Transferul parametrilor

- Problema: citirea a două numere n și m de la tastatură într-o funcție și vizibilitatea valorilor la revenire din apel
- Transferul prin valoare sau prin referință
  - Transmiterea prin valoare: modificările aduse parametrilor au efect doar în interiorul funcției
  - Transmiterea prin referință: se transmite o adresă, modificările asupra datelor de la adresa respectivă sunt vizibile și din afara funcției

Transferul parametrilor

- Problema: citirea a două numere n și m de la tastatură într-o funcție și vizibilitatea valorilor la revenire din apel
- Transferul prin valoare sau prin referință
  - Transmiterea prin valoare: modificările aduse parametrilor au efect doar în interiorul funcției
  - Transmiterea prin referință: se transmite o adresă, modificările asupra datelor de la adresa respectivă sunt vizibile și din afara funcției

Transferul parametrilor

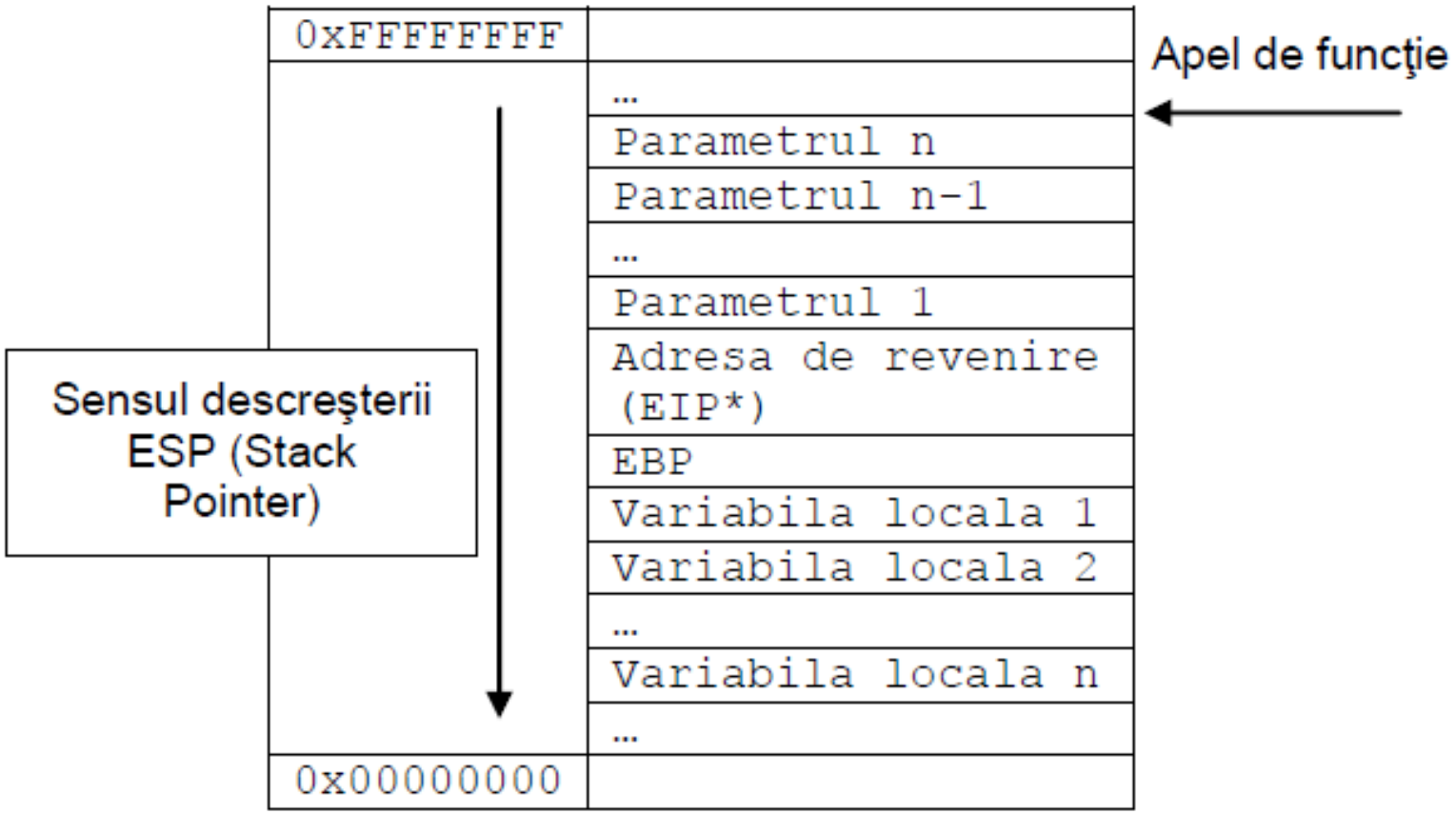
- Problema: citirea a două numere n și m de la tastatură într-o funcție și vizibilitatea valorilor la revenire din apel
- Transferul prin valoare sau prin referință
  - Transmiterea prin valoare: modificările aduse parametrilor au efect doar în interiorul funcției
  - Transmiterea prin referință: se transmite o adresă, modificările asupra datelor de la adresa respectivă sunt vizibile și din afara funcției

Transferul parametrilor

- Problema: citirea a două numere n și m de la tastatură într-o funcție și vizibilitatea valorilor la revenire din apel
- Transferul prin valoare sau prin referință
  - Transmiterea prin valoare: modificările aduse parametrilor au efect doar în interiorul funcției
  - Transmiterea prin referință: se transmite o adresă, modificările asupra datelor de la adresa respectivă sunt vizibile și din afara funcției

Transferul parametrilor

- Recapitulare: stack și heap



Transferul parametrilor

- Problema: citirea a două numere n și m de la tastatură într-o funcție și vizibilitatea valorilor la revenire din apel
- Soluția:
  - Parametri actuali: adrese de variabile
  - Parametri formali: pointeri
- Reamintire `scanf()`

Problema

- Să se scrie prototipul, definiția și apelul unei funcții pentru citirea unui vector de numere întregi de la tastatură (datele citite să fie vizibile în afara funcției)
  - 1. Funcția NU va aloca dinamic memoria
  - 2. Funcția VA aloca dinamic memoria
- Să se implementeze și funcția pentru dealocarea memoriei

Problema

- Să se scrie prototipul, definiția și apelul unei funcții pentru citirea unei matrici de numere întregi nxm de la tastatură (datele citite să fie vizibile în afara funcției) - funcția va aloca dinamic memoria
- Să se implementeze și funcția pentru dealocarea memoriei

Variabile locale și globale

- Variabilele declarate în corpul unei funcții sunt variabile locale
- Cele declarate în afara corpului sunt variabile globale
- Aplicațiile trebuie să limiteze numărul variabilelor globale pentru limitarea accesărilor accidentale - datele să fie încapsulate, acces restricționat
- Variabile statice locale
- Variabile statice globale