PROGRAMARE PROCEDURALĂ

Bogdan Alexe
bogdan.alexe@fmi.unibuc.ro

Secția Informatică, anul I,
2018-2019

Cursul 14

Evaluarea activităților didactice pe semestrul 1

- studenții au posibilitatea să evalueze activitățile cadrelor didactice
- evaluare = completarea unui chestionar pentru fiecare disciplină (curs/ seminar/laborator) ce conține 11 întrebări = 3-5 minute
- evaluările au caracter anonim, fiecare student se va loga folosind un token nenominal de unică folosință primit de la secretariat de către șeful de grupă și distribuit apoi membrilor grupei.
- evaluarea se desfășoară în perioada 9 – 20 ianuarie
- concluziile din evaluarea de anul trecut vor fi făcute publice într-un raport (cel mai probabil în ianuarie)

Sesiune activa ani terminali | Sesiune activa ani neterminali | Numarul de studenti care au evaluat

Show 10 entries

Grupa	Numar Studenti
131	5
132	16
133	13
134	9
135	2
141	17

Organizare examen

- Vrem să afișăm situația voastră (notă laborator + bonus seminar) până duminică, 27 ianuarie, cu o săptămână înainte de examenul final din 4 februarie 2019;

- Examenul final scris va fi pe 4 februarie 2019 (cel mai probabil ora 9:00) în toate cele 4 amfiteatre; durata − 2 ore; să aveți cu voi un act de identitate, pix, fără telefoane mobile, fără foile voastre;

- Luni, 11 februarie, intenționăm (nu știu dacă o să reușim, avem de corectat aproximativ 150 de lucrări) să facem publice notele. Cei nemulțumiți de nota la scris își pot vedea lucrările în ziua de marți, 12 februarie. O să comunic exact datele/orele șefilor de grupă.

Recapitulare – cursul trecut

1. Programare generică

Programa cursului

Introducere

- Algoritmi
- Limbaje de programare.

Fundamentele limbajului C

- Introducere în limbajul C. Structura unui program C.
- Tipuri de date fundamentale. Variabile. Constante. Operatori. Expresii. Conversii.
- Tipuri derivate de date: tablouri, șiruri de caractere, structuri, uniuni, câmpuri de biți, enumerări, pointeri
- Instrucțiuni de control
- Directive de preprocesare. Macrodefiniții.
- Funcții de citire/scriere.
- Etapele realizării unui program C.

Fișiere text

- Funcții specifice de manipulare.

Fișiere binare

- Funcții specifice de manipulare.

Funcții (1)

- Declarare și definire. Apel. Metode de transmitere a parametrilor. Pointeri la funcții.

Tablouri și pointeri

- Aritmetica pointerilor
- Legătura dintre tablouri și pointeri
- Alocarea dinamică a memoriei
- Clase de memorare

Șiruri de caractere

- Funcții specifice de manipulare.

Structuri de date complexe și autoreferite

- Definire și utilizare

Funcții (2)

- Funcții cu număr variabil de argumente.
- Preluarea argumentelor funcției main din linia de comandă.
- Programare generică.

Recursivitate

Cuprinsul cursului de azi

1. Recursivitate

2. Model de subiect de examen

Recursivitate

- recursivitate = capacitatea unei funcții de a se auto-apela
- studiem mecanismul recursivității, ce se întâmplă când o funcție se auto-apelează (nu studiem recursivitatea ca tehnică de programare)
- corespondentul din matematică al recursivității este recurența

Recursivitate - exemplu

1. Definiția factorialului unui număr natural n

$$n! = \begin{cases} 1, \text{ dacă } n=0 \\ n*(n-1)!, \text{ dacă } n>=1 \end{cases}$$

4! = 4*3! = 4 * 6 = 24
3! = 3*2! = 3 * 2 = 6
2! = 2*1! = 2 * 1 = 2
1! = 1*0! = 1 * 1 = 1
0! = 1 (stop)

Adâncimea recursivității

Altă definiție: $$n! = \frac{(n+1)!}{n+1}$$

Definiție inutilă, nu se oprește relația de recurență.

Recursivitate - exemplu

1. Definiția factorialului unui număr natural n

$$n! = \begin{cases} 1, \text{ dacă } n=0 \\ n*(n-1)!, \text{ dacă } n>=1 \end{cases}$$

```
int factorial(int n)
{
    if (n==0) return 1; //conditia de oprire
    return n*factorial(n-1);//recursivitate
}
```

- ce se întâmplă în stivă pentru apelul t = factorial(4) ?
- în stivă, fiecare apel se așează deasupra apelului precedent.
- se salvează un context de apel.

Recursivitate - stivă

- ce se întâmplă în stivă pentru apelul t = factorial(4) ?
- în stivă, fiecare apel se așează deasupra apelului precedent.
- se salvează un context de apel:

1. adresa de revenire
2. copii ale valorile parametrilor efectivi
3. valorile variabilelor locale
4. copii ale regiștrilor
5. valoarea returnată

STIVĂ

A_5	0	-	-	1
A_4	1	-	-	1
A_3	2	-	-	2
A_2	3	-	-	6
A_1	4	-	-	24

Adresa de revenire | Valoare parametri efectivi | Valoare Variabile locale | Regiștri | Valoare returnată

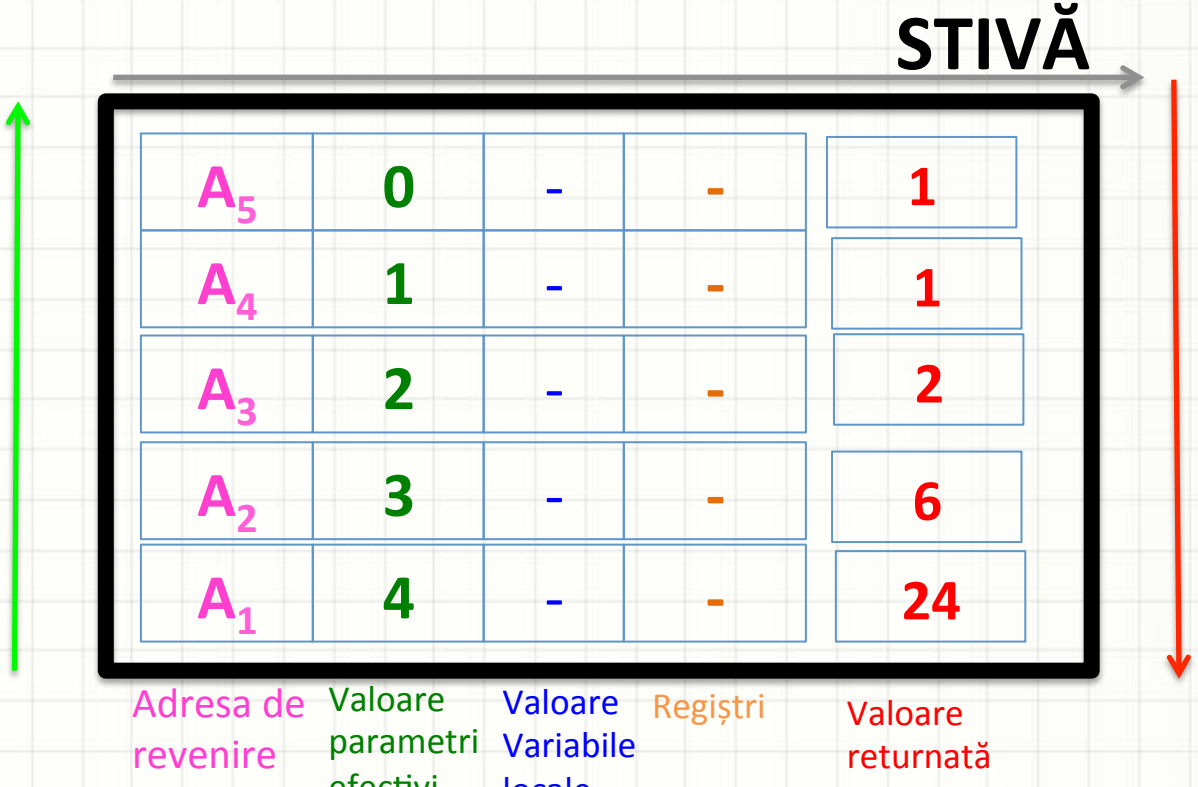
```
t = factorial(4);
```

A_1

Recursivitate - exemplu

Trei faze în execuție:

1. expandarea recursivității (crește stiva)
2. condiția de oprire a recursivității
3. restaurarea stivei (a contextului de apel)



t = factorial(4);

A_1

Recursivitate - exemplu

2. Citesc numere întregi până la întâlnirea lui 0 și le afișez în ordine inversă.

exempluRecusivitate1.c

```c
#include <stdio.h>

void citesteAfiseaza()
{
    int x;
    scanf("%d", &x);
    if(x)
        citesteAfiseaza();
    printf("%d ",x);
    return;
}

int main()
{
    citesteAfiseaza();
    printf("\n");
    return 0;
}
```

```
Bogdan-Alexes-MacBook-Pro:curs13 bogdan$ gcc exempluRecusivitate1.c
Bogdan-Alexes-MacBook-Pro:curs13 bogdan$ ./a.out
10
20
30
0
0 30 20 10
```

Ce se întâmplă dacă în loc de int x am int static x (linia 5)?

Recursivitate - exemplu

2. Citesc numere întregi până la întâlnirea lui 0 și le afișez în ordine inversă.

exempluRecusivitate2.c

```c
#include <stdio.h>

void citesteAfiseaza()
{
    int static x;
    scanf("%d", &x);
    if(x)
        citesteAfiseaza();
    printf("%d ",x);
    return;
}

int main()
{
    citesteAfiseaza();
    printf("\n");
    return 0;
}
```

```
Bogdan-Alexes-MacBook-Pro:curs13 bogdan$ gcc exempluRecusivitate2.c
Bogdan-Alexes-MacBook-Pro:curs13 bogdan$ ./a.out
10
20
30
0
0 0 0 0
```

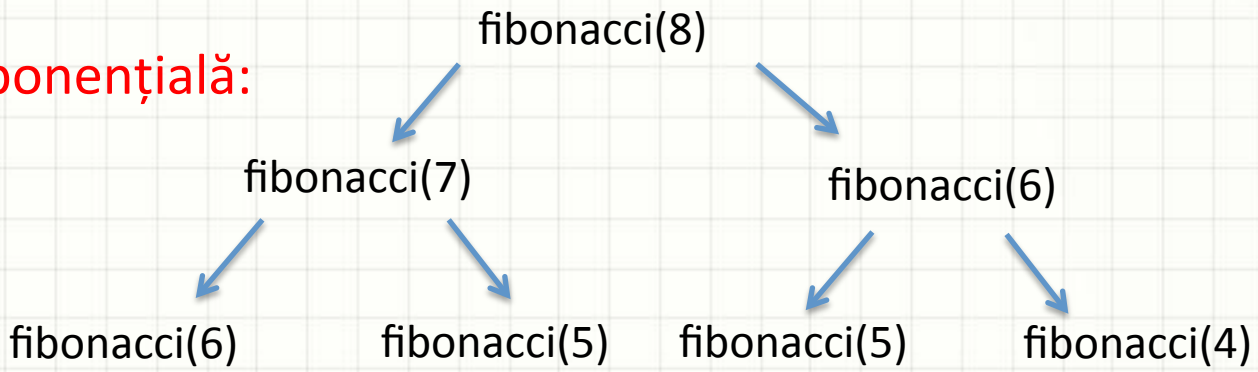
x nu mai e alocat pe stiva funcției, se păstreaza ultima valoare.

Recursivitate - exemplu

3. Șirul lui Fibonacci: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$, dacă n>=2

```
int fibonacci(int n)
{
    if (n<=1)
        return n;
    return fibonacci(n-1) + fibonacci(n-2);
}
```

Complexitate exponențială:



Recursivitate - exemplu

3. Șirul lui Fibonacci: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$, dacă n>=2

exempluRecusivitate3.c

```c
#include <stdio.h>

int fibonacci(int n)
{
    static int nrApeluri = 0;
    printf("Am ajuns la apelul %d\n",++nrApeluri);

    if(n<=1)
        return n;
    return fibonacci(n-2) + fibonacci(n-1);
}

int main()
{
    int n = 20;
    printf("F_%d = %d \n",n,fibonacci(n));
    return 0;
}
```

```
Am ajuns la apelul 21890
Am ajuns la apelul 21891
F_20 = 6765
```

Putem număra câte apeluri se efectuează.

Recursivitate - exemplu

4. Afisarea unei matrice

exempluRecusivitate4.c

```c
#include <stdio.h>

void afisare(int t[4][4], int m, int n, int i, int j)
{
    if(i < m)
    {
        if(j < n)
        {
            printf("%d ", t[i][j]);
            afisare(t, m, n, i, j+1);
        }
        else
        {
            printf("\n");
            afisare(t, m, n, i+1, 0);
        }
    }
}


int main()
{
    int mat[4][4], i, j;
    for(i = 0; i <4; i++)
        for (j = 0; j < 4; j++)
            mat[i][j] = i+j;
    afisare(mat, 4, 4, 0, 0);
    return 0;
}
```

Recursivitate - exemplu

4. Afisarea unei matrice

exempluRecusivitate4.c

```c
#include <stdio.h>

void afisare(int t[4][4], int m, int n, int i, int j)
{
    if(i < m)
    {
        if(j < n)
        {
            printf("%d ", t[i][j]);
            afisare(t, m, n, i, j+1);
        }
        else
        {
            printf("\n");
            afisare(t, m, n, i+1, 0);
        }
    }
}


int main()
{
    int mat[4][4], i, j;
    for(i = 0; i <4; i++)
        for (j = 0; j < 4; j++)
            mat[i][j] = i+j;
    afisare(mat, 4, 4, 0, 0);
    return 0;
}
```

```
Bogdan-Alexes-MacBook-Pro:curs13 bogdan$ gcc exempluRecusivitate4.c
Bogdan-Alexes-MacBook-Pro:curs13 bogdan$ ./a.out
0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6
```

Cuprinsul cursului de azi

1. Recursivitate

2. Model de subiect de examen

UNIVERSITATEA DIN BUCUREŞTI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
DEPARTAMENTUL DE INFORMATICĂ
DOMENIUL DE STUDII INFORMATICĂ
SERIILE 13 ŞI 14

EXAMEN LA DISCIPLINA "PROGRAMARE PROCEDURALĂ"
— SESIUNEA IANUARIE/FEBRUARIE 2017 —

Subiectul I – 2 puncte

a) Scrieţi o funcţie care să returneze un tablou unidimensional alocat dinamic format din valorile pare aflate într-un tablou unidimensional v având n elemente de tip întreg, precum şi numărul acestora. (1 punct)

b) Considerăm definite următoarele 4 funcţii:
   - $void\ add(int^*\ v, int\ n, int\ x)$ – adună valoarea x la fiecare element al tabloului unidimensional v format din n numere întregi;
   - $void\ multiply(int^*\ v, int\ n, int\ x)$ – înmulţeşte cu x fiecare element al tabloului unidimensional v format din n numere întregi;
   - $void\ sort(int^*\ v, int\ n)$ – sortează crescător tabloul unidimensional v format din n numere întregi;
   - $void\ display(int^*\ v, int\ n)$ – afişează pe ecran tabloul unidimensional v format din n numere întregi.

   Folosind doar apeluri utile ale funcţiei definite la punctul a) şi ale celor 4 funcţii precizate mai sus, scrieţi o funcţie care să afişează în ordine descrescătoare numerele impare dintr-un tabloul unidimensional v format din n numere întregi. (1 punct)

Observaţie: Nu este permisă utilizarea unor variabile globale!

Subiectul nr. II - 1 punct

Scrieți o funcție care primește ca parametru un număr natural nenul $n \geq 2$ și returnează un tablou bidimensional triunghiular alocat dinamic și construit după următoarele reguli:

- prima coloană conține numerele de la 1 la n, în ordine crescătoare;
- ultima linie conține numerele de la 1 la n, în ordine descrescătoare;
- orice alt element este egal cu suma vecinilor săi de la vest, sud și sud-vest.

Exemplu: Pentru $n = 4$ funcția trebuie să returneze următorul tablou:

```
1
2   15
3   10   15
4    3    2    1
```

Subiectul nr. III - 2 puncte

a) Scrieți o funcție cu număr variabil de parametri care să returneze minimul dintr-un șir de numere întregi. (1 punct)

b) Scrieți o funcție cu 4 parametri de tip întreg a, b, c și d care verifică dacă aceștia îndeplinesc condiția $a \geq b \geq c \geq d$ sau nu, folosind apeluri utile ale funcției definite anterior. (1 punct)

Subiectul IV – 2 puncte

Fişierul text *cuvinte.in* conţine pe prima linie un cuvânt w format din $n \geq 1$ litere mici ale alfabetului englez, iar pe următoarele linii un text în care cuvintele sunt despărţite prin spaţii şi semnele de punctuaţie uzuale. Realizaţi un program care să scrie în fişierul text *cuvinte.out* toate cuvintele din fişierul *cuvinte.in* care sunt anagrame ale cuvântului w sau mesajul "*Imposibil*" dacă în fişierul de intrare nu există nici un cuvânt cu proprietatea cerută. Două cuvinte sunt anagrame dacă sunt formate din exact aceleaşi litere, indiferent de ordinea lor.

Observaţie: Se vor utiliza funcţii pentru manipularea şirurilor de caractere din biblioteca `string.h`!

Subiectul V – 2 puncte

a) Scrieţi o funcţie care să încarce într-un tablou unidimensional alocat dinamic conţinutul unui fişier binar în care sunt memorate numere întregi, să sorteze descrescător tabloul respectiv folosind funcţia `qsort` din `stdlib.h` şi apoi să-l afişeze pe ecran. Funcţia va avea ca parametru numele fişierului binar. (1 punct)

b) Scrieţi o funcţie care să modifice semnul fiecărui număr întreg aflat într-un fişier binar (un număr negativ va deveni pozitiv şi invers). Funcţia va avea ca parametru numele fişierului binar. (1 punct)

Observații

- modelul de subiect prezentat anterior este **un model**;

- aveți 7 seminarii, fiecare (mai puțin primul) conținând probleme de tipul celor ce se vor da la examenul final scris: (1) seminar recapitulativ, (2) operatori pe biți, (3) structuri, uniuni, enumerări, câmpuri de biți, (4) fișiere text și binare, (5) pointeri la funcții și alocare dinamică, (6) șiruri de caractere, (7) programare generica + funcții cu număr variabil de parametri, recursivitate;

- veți primi la examen probleme asemănătoare cu cele discutate la curs, seminar, laborator -> multe modele la dispoziție

TE DUCI LA EXAMEN SPERÂND
CĂ TE VOR AJUTA COLEGII.

TE DUCI LA EXAMEN SPERÂND CĂ TE VOR AJUTA COLEGII. COINCIDENȚĂ: ȘI EI SE GÂNDEAU LA ACELAȘI LUCRU.