PROGRAMARE PROCEDURALĂ

Bogdan Alexe
bogdan.alexe@fmi.unibuc.ro

Secția Informatică, anul I,
2018-2019

Cursul 11

Proiect – criptare + decriptare

- am primit mai multe mesaje de la studenți că nu se potrivește imaginea noastră criptată cu a voastră. Trebuie să verificăm acest lucru, eventual o să actualizez arhiva pe moodle săptămâna aceasta (pun un mesaj pe moodle)

Proiect – criptare + decriptare

Ideea este urmatoarea:

1. Fie I o imagine de dimensiune 2x2 pixeli => forma sa liniarizata este L=(P[0][0], P[0][1], P[1][0], P[1][1]), deci avem nevoie de 2*W*H-1 = 2*2*2-1 = 7 numere aleatoare pentru intreg procesul de criptare
2. Generam 7 numere aleatoare folosind algoritmul XORSHIFT32 cu valoarea initiala R0 din cheia secreta si obtinem numerele (213, 18, 1135, 714, 6, 800, 2018) (exemplul este luat la intamplare, fara nicio legatura cu XORSHIFT32)
3. Pentru a genera o permutare aleatoare cu W*H = 2*2 = 4 elemente, algoritmul lui Durstenfeld necesita W*H-1 valori, deci le vom folosi pe primele 3, astfel:
    1. initializam σ = (0,1,2,3)
    2. pentru k = 3 (ultimul indice) calculam r = 213%(k + 1) = 1 (am impartit la k+1 deoarece algoritmul lui Durstenfeld necesita un indice aleator r cuprins intre 0 si indicele curent k) = > interschimbam σ(k=3) cu σ(r=1) => σ = (0,3,2,1)
    3. pentru k = 2 calculam r = 18%(k+1) = 0 = > interschimbam σ(k=2) cu σ(r=0) => σ = (2,3,0,1)
    4. pentru k = 1 calculam r = 1135%(k+1) = 1 = > interschimbam σ(k=1) cu σ(r=1) => σ = (2,3,0,1)
4. Am obtinut permutarea aleatoare σ = (2,3,0,1) si o aplicam asupra imaginii liniarizate L pentru a obtine imaginea intermediara permutata L' astfel:
    1. L'[σ[0]] = L[0] => L'[2] = L[0]
    2. L'[σ[1]] = L[1] => L'[3] = L[1]
    3. L'[σ[2]] = L[2] => L'[0] = L[2]
    4. L'[σ[3]] = L[3] => L'[1] = L[3]
5. Astfel, am obtinut imaginea intermediara L'=(L[2], L[3], L[0], L[1]) = (P[1][0], P[1][1], P[0][0], P[0][1])
6. Asupra fiecarui pixel al imaginii L' aplicam formulele de substitutie folosind cele 4 valori aleatoare ramase (714, 6, 800, 2018) prin metoda de XOR-are prezentata in partea de preliminarii (fiecare octet fara semn RGB al unui pixel se XOR-eaza cu cate un octet fara semn al unui intreg fara semn aleator, mai putin octetul cel mai semnificativ al numarului aleator)

Test de laborator

- sâmbătă, 15 decembrie, seria 13 între 9-11, seria 14 între 12-14;
- fără materiale, fără net;
- subiect unic;
- afișăm repartizarea pe laboratoare vineri seara pe moodle;
- dacă nu puteți da testul dintr-un motiv bun mă anunțați până astă seară. Cei care nu participă la test iau 0 puncte.

- preferință pentru examenul final: verificare sau examen?

TEST LA LABORATOR – "PROGRAMARE PROCEDURALĂ"
MODEL DE SUBIECT

Subiectul I – 3 puncte

Numim **tablou bidimensional alternant** un tablou în care elementele de pe coloane pare sunt egale cu numărul liniei pe care se află, iar cele de pe coloane impare sunt egale cu numărul coloanei pe care se află.

a) Scrieți o funcție cu numele *linie*, cu doi parametri, *n* și *k*, numere naturale, care construiește un tablou unidimensional cu *n* elemente, în care valorile de pe poziții pare sunt egale cu *k*, iar celelalte sunt egale cu poziția pe care se află (prima poziție este 0). Funcția va returna adresa care permite accesul la elementele tabloului, alocat dinamic. (1 punct)

b) Scrieți o funcție care citește de la tastatură două numere naturale, *nrl* și *nrc*, apoi, folosind apeluri utile ale funcției *linie*, construiește un tablou bidimensional alternant cu *nrl* linii și *nrc* coloane. Valorile citite, precum și adresa care permite accesul la elementele tabloului, alocat dinamic, sunt transmise ca parametri. (1 punct)

c) Scrieți un program care, utilizând apeluri utile ale funcției precizate mai sus, construiește un tablou bidimensional alternant, apoi scrie în fișierul binar *tablou.out* cele două numere reprezentând dimensiunile tabloului, apoi elementele acestuia. (1 punct)

Subiectul II – 6 puncte

Fișierul text *temperaturi.txt* conține pe fiecare linie numele unui oraș (un singur cuvânt) și temperatura medie anuală înregistrată în orașul respectiv (număr real pozitiv cu cel mult două zecimale), despărțite printr-un spațiu.

a) Scrieți pe prima linie a fișierului text *maxime.txt* temperatura medie anuală maximă înregistrată, iar pe următoarele linii numele orașelor în care aceasta a fost atinsă. (2 puncte)

b) Afișați pe ecran media aritmetică a temperaturilor medii anuale înregistrate în orașele respective. (2 puncte)

c) Sortați orașele în ordinea descrescătoare a temperaturilor medii anuale, iar în cazul unor temperaturi medii egale în ordine alfabetică. Scrieți in fișierul text *temp_sort.txt* ierarhia astfel obținută. (2 puncte)

NOTĂ:

1. În rezolvarea subiectelor I nu este permisă utilizarea unor variabile globale.
2. Pentru sortarea tablourilor se va utiliza funcția `qsort` din biblioteca `stdlib.h`.
3. Rezolvările corecte care nu respectă restricțiile indicate (rezolvarea unor cerințe fără a folosi funcții, utilizarea unor variabile globale, utilizarea unor tablouri alocate static, neutilizarea funcției `qsort` pentru sortarea unui tablou etc.) vor primi punctaje parțiale.
4. Ambele subiecte trebuie să fie rezolvate utilizând strict limbajul C standard, ci nu limbajul C++!
5. Se acordă 1 punct din oficiu.

Recapitulare – cursul trecut

1. Clase de memorare

2. Șiruri de caractere − funcții specifice de manipulare

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. Tablouri de șiruri de caractere
2. Preluarea argumentelor funcției main din linia de comandă
3. Funcții cu număr variabil de argumente
4. Declarații complexe

Funcțiile din string.h

`char* strcpy(char* d,const char* s)`	copiază şirul **s** în **d**, inclusiv **'\0'**, întoarce **d**
`char* strncpy(char* d,const char* s, int n)`	copiază **n** caractere din şirul **s** în **d**, completând eventual cu **'\0'**, întoarce **d**
`char* strcat(char* d,const char* s)`	concatenează şirul **s** la sfârşitul lui **d**, întoarce **d**
`char* strncat(char* d,const char* s, int n)`	concatenează cel mult **n** caractere din şirul **s** la sfârşitul lui **d**, completând cu **'\0'**, întoarce **d**
`int strcmp(const char* d, const char* s)`	compară şirurile **d** şi **s**, întoarce **-1** dacă **d<s**, **0** dacă **d==s** şi **1** dacă **d>s**
`int stricmp(const char* d, const char* s)`	compară şirurile **d** şi **s** (ca şi **strcmp()**) fără a face distincţie între litere mari şi mici
`int strncmp(const char* d, const char* s, int n )`	similar cu **strcmp()**, cu deosebirea că se compară cel mult **n** caractere
`char* strchr(const char* d,char c)`	caută caracterul **c** în şirul **d**; întoarce un pointer la prima apariţie a lui **c** în **d**, sau **NULL**
`char* strrchr(const char* d,char c)`	întoarce un pointer la ultima apariţie a lui **c** în **d**, sau **NULL**
`char* strstr(const char* d, const char* s)`	întoarce un pointer la prima apariţie a subşirului **s** în **d**, sau **NULL**
`char* strpbrk(const char* d, const char* s)`	întoarce un pointer la prima apariţie a unui caracter din subşirul **s** în **d**, sau **NULL**

Funcțiile din string.h

`int strspn(const char* d, const char* s)`	întoarce lungimea prefixului din **d** care conţine numai caractere din **s**
`int strcspn(const char* d, const char* s)`	întoarce lungimea prefixului din **d** care conţine numai caractere ce nu apar în **s**
`int strlen(const char* s)`	întoarce lungimea lui **s** (`'\0'` nu se numără)
`void* memcpy(void* d, const void* s,int n)`	copiaza **n** octeţi din **s** în **d**; întoarce **d**
`void* memmove(void* d, const void* s,int n)`	ca şi **memcopy**, folosită daca **s** şi **d** se întrepătrund
`void* memset(void* d,const int c, int n)`	copiază caracter **c** în primele **n** poziţii din **d**
`int memcmp(const void* d, const void* s,int n)`	compară zonele adresate de **s** şi **d**
`char* strtok(const char* d, const char* s)`	caută în **d** subşirurile delimitate de caracterele din **s**;primul apel întoarce un pointer la primul subşir din **d** care nu conţine caractere din **s** următoarele apeluri se fac cu primul argument **NULL**, întorcându-se de fiecare dată un pointer la următorul subşir din **d** ce nu conţine caractere din **s**; în momentul în care nu mai există subşiruri, funcţia întoarce **NULL**

Tablouri de șiruri de caractere

- **un șir de caractere (string)** este un tablou unidimensional cu elemente de tip char terminat cu caracterul '\0' (NUL)
- se poate reprezenta ca:
  - tablou de caractere:
    - char sir1[10] = "exemplu"; //se aloca 10 octeti
  - pointer la caractere:
    - char *sir4; //se aloca memorie numai pentru pointer

- un tablou de șiruri = un tablou cu elemente șiruri de caractere
- se poate reprezenta ca:
  - tablou de tablouri de caractere = tablou bidimensional
    - char nume[4][20];
  - tablou de pointeri la caractere
    - char* nume[4];

Tablouri de șiruri de caractere

tablouSiruri.C

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char nume[4][10] = {"Miruna","Claudiu","Ionut","Paul"};
    int i;
    for(i=0;i<4;i++)
        printf("%s \n",nume[i]);

    printf("\n");

    nume[3][0] = 'R';
    for(i=0;i<4;i++)
        printf("%s \n",nume[i]);

    printf("\n dimensiunea lui nume este: %lu \n",sizeof(nume));

    return 0;
}
```

```
Miruna
Claudiu
Ionut
Paul

Miruna
Claudiu
Ionut
Raul

 dimensiunea lui nume este: 40
```

Tablouri de șiruri de caractere

tablouSiruri2.C

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char* nume[4] = {"Miruna","Claudiu","Ionut","Paul"};
    int i;
    for(i=0;i<4;i++)
        printf("%s \n",nume[i]);

    printf("\n");

    //nume[3][0] = 'R'; NU AM VOIE SA FAC ASTA

    printf("\n dimensiunea lui nume este: %lu \n",sizeof(nume));

    return 0;
}
```

```
Miruna
Claudiu
Ionut
Paul


 dimensiunea lui nume este: 32
```

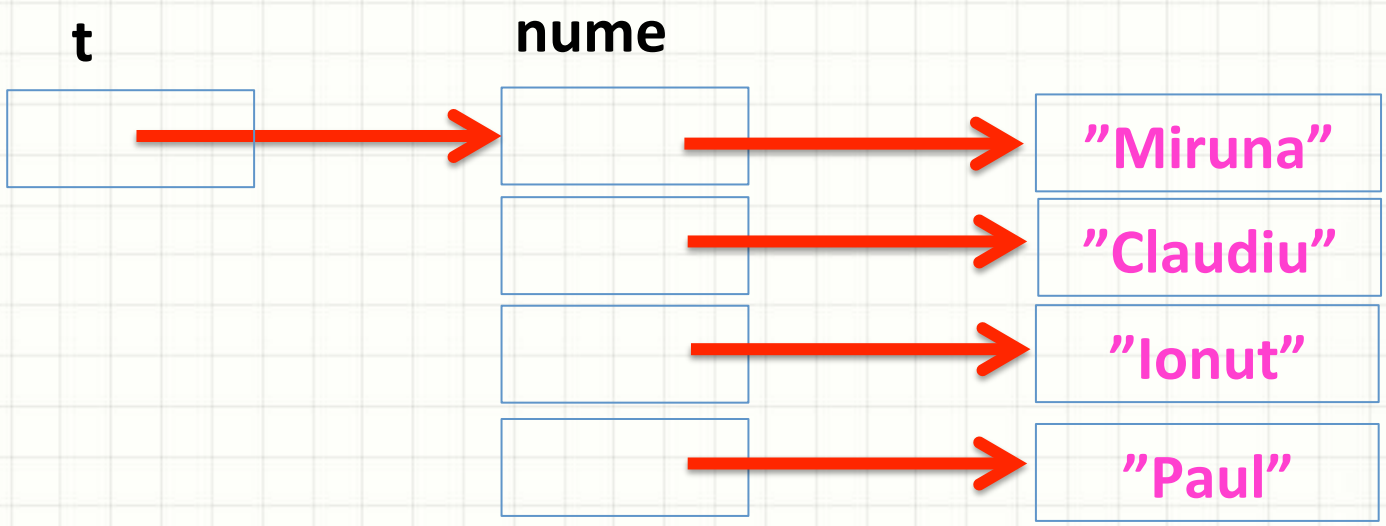
Tablouri de șiruri de caractere

tablouSiruri3.C

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char* nume[4] = {"Miruna","Claudiu","Ionut","Paul"};
    char **t = nume;
    int i;
    for(i=0;i<4;i++)
        printf("%s \n",t[i]);

    printf("\n");

    printf("\n dimensiunea lui t este: %lu \n",sizeof(t));

    return 0;
}
```

```
Miruna
Claudiu
Ionut
Paul


 dimensiunea lui t este: 8
```

Tablouri de șiruri de caractere



Cuprinsul cursului de azi

1. Tablouri de șiruri de caractere
2. Preluarea argumentelor funcției main din linia de comandă
3. Funcții cu număr variabil de argumente
4. Declarații complexe

Preluarea argumentelor funcției main din linia de comandă

- un program executabil (comandă) poate fi lansat în execuţie de către interpretorul de comenzi al sistemului de operare.
- de exemplu, programul **reminder1** care afişează la terminal un mesaj:

```
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ gcc -o reminder1 reminder1.c
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ ./reminder1

 Vacanta incepe pe 21 decembrie!
```

reminder1.c

```c
#include <stdio.h>
#include <string.h>

int main()
{

    printf("\n Vacanta incepe pe 21 decembrie! \n\n");
    return 0;
}
```

- program fără parametri, lansat în execuţie prin:
  >./reminder1

Preluarea argumentelor funcției main din linia de comandă

- un program poate avea şi parametri, care apar după numele comenzii şi sunt separaţi prin spaţii libere.

- de exemplu, programul **reminder2** ar putea avea un parametru şir de caractere, fiind lansat în execuţie, în acest caz prin:

> ./reminder2 parametru

```
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ gcc -o reminder2 reminder2.c
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ ./reminder2 9

 Vacanta incepe pe 21 decembrie! Au mai ramas 9 zile!
```

Preluarea argumentelor funcției main din linia de comandă

- un program poate avea şi parametri, care apar după numele comenzii şi sunt separaţi prin spaţii libere.

- programul poate avea acces la parametrii liniei de comandă, dacă funcţia **main()** prezintă argumente:

```
int main(int argc, char *argv[])
{ … }
```

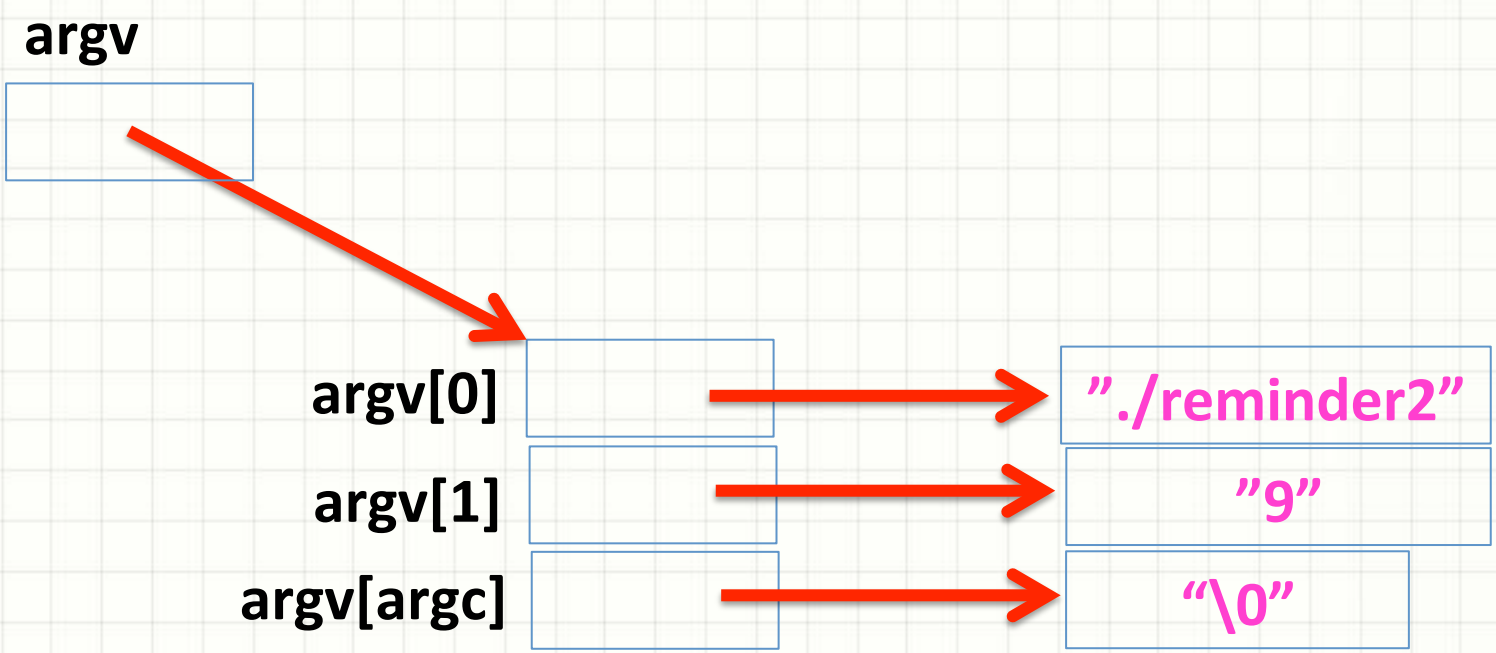
- primul argument, **argc** (*argumentul contor*) este o variabilă de tip întreg care reprezintă *numărul de parametri ai programului/comenzii.*

- al doilea argument, **argv** (*argumentul vector*) este un pointer la un tablou de pointeri la şiruri de caractere, care conţin argumentele (fiecare element al tabloului = un pointer la un şir de caractere = un argument).

Preluarea argumentelor funcției main din linia de comandă

- primul argument, **argc** (*argumentul contor*)
- al doilea argument, **argv** (*argumentul vector*)
- **argv[0]** conţine întotdeauna *numele programului*;
- **argv[1]** conţine *primul parametru* ca șir de caractere;
- **argv[2]** conţine *al doilea parametru* ca șir de caractere;
- **argv[argc-1]** conţine *ultimul parametru* ca șir de caractere;
- **argv[argc]** va fi un pointer la un şir vid (**NULL**);
- pentru o comandă fără parametri **argc=1**, iar o comandă cu 2 parametri va avea **argc=3**.

Preluarea argumentelor funcției main din linia de comandă




```
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ gcc -o reminder2 reminder2.c
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ ./reminder2 9

 Vacanta incepe pe 21 decembrie! Au mai ramas 9 zile!
```

Preluarea argumentelor funcției main din linia de comandă

reminder2.c

```c
#include <stdio.h>
#include <string.h>


int main(int argc, char *argv[])
{

    char mesaj[1000];
    strcpy(mesaj,"\n Vacanta incepe pe 21 decembrie!");

    if (argc==2)
    {
        strcat(mesaj," Au mai ramas ");
        strcat(mesaj, *(argv+1));
        strcat(mesaj, " zile! \n\n");
    }

    printf("%s",mesaj);
    return 0;
}
```

```
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ gcc -o reminder2 reminder2.c
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ ./reminder2 9

 Vacanta incepe pe 21 decembrie! Au mai ramas 9 zile!
```

Preluarea argumentelor funcției main din linia de comandă

reminder3.c

```c
#include <stdio.h>
#include <string.h>


int main(int argc, char *argv[])
{

    char mesaj[1000];
    strcpy(mesaj,"\n Vacanta incepe pe 21 decembrie!");

    for(int i = 1; i< argc; i++)
    {
        strcat(mesaj, argv[i]);
        strcat(mesaj, " ");
    }
    strcat(mesaj," \n\n");
    printf("%s",mesaj);

    return 0;
}
```

```
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ gcc -o reminder3 reminder3.c
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ ./reminder3 "Pana atunci mai e testul de PP." "Bafta! "

 Vacanta incepe pe 21 decembrie!Pana atunci mai e testul de PP. Bafta!
```

Preluarea argumentelor funcției main din linia de comandă

reminder3.c

```c
#include <stdio.h>
#include <string.h>


int main(int argc, char *argv[])
{

    char mesaj[1000];
    strcpy(mesaj,"\n Vacanta incepe pe 21 decembrie!");

    for(int i = 1; i< argc; i++)
    {
        strcat(mesaj, argv[i]);
        strcat(mesaj, " ");
    }
    strcat(mesaj," \n\n");
    printf("%s",mesaj);

    printf("argc = %d \n",argc);
    return 0;
}
```

```
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ ./reminder3 "Pana atunci mai e testul de PP." "Bafta! "

 Vacanta incepe pe 21 decembrie!Pana atunci mai e testul de PP. Bafta!

argc = 3
```

Preluarea argumentelor funcției main din linia de comandă

reminder3.c

```c
#include <stdio.h>
#include <string.h>


int main(int argc, char *argv[])
{

    char mesaj[1000];
    strcpy(mesaj,"\n Vacanta incepe pe 21 decembrie!");

    for(int i = 1; i< argc; i++)
    {
        strcat(mesaj, argv[i]);
        strcat(mesaj, " ");
    }
    strcat(mesaj," \n\n");
    printf("%s",mesaj);

    printf("argc = %d \n",argc);
    return 0;
}
```

```
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ ./reminder3 Pana atunci mai e testul de PP. Bafta!

 Vacanta incepe pe 21 decembrie!Pana atunci mai e testul de PP. Bafta!

argc = 9
```

Cuprinsul cursului de azi

1. Tablouri de șiruri de caractere
2. Preluarea argumentelor funcției main din linia de comandă
3. Funcții cu număr variabil de argumente
4. Declarații complexe

Funcții cu număr variabil de argumente

- funcții cu număr variabil de argumente utilizate de către voi până acum:
  - printf, fprintf
  - scanf, fscanf

```
int a, b, c;
scanf("%d %d", &a, &b);
printf("a = %d \n b = d \n", a ,b);
scanf("%d", &c);
printf("\n\n");
printf("c = %d\n", c);
```

Funcții cu număr variabil de argumente

- probleme:
  - nu se cunoaște numărul parametrilor funcției
  - nu se cunoaște tipul parametrilor
- header-ul stdarg.h cuprinde:
  - definiția unui tip de date specializat (va_list) dedicat manipulării listelor cu număr variabil de parametri
  - macro-uri (va_start, va_arg, va_end) care realizează operații pe acest tip de date
  - macro = fragment de cod căruia i se asociază un nume, la preprocesare se înlocuiește numele cu fragmentul de cod.

Funcții cu număr variabil de argumente

- **sintaxa:**

tip_rezultat nume_functie(tip_argument nume_argument, …)

unde (***cel putin***) primul argument este întotdeauna fix, vizibil, restul argumentelor fiind declarate prin *cele trei puncte*, (...) – *mecanismul elipsă*

- **exemplu:** funcție ce calculează suma a n numere întregi

int suma(int n,…);

suma(4,1,2,1,1) -> 5; suma(5,1,2,1,1,3) ->8

Funcții cu număr variabil de argumente

- macro-urile din stdarg.h:
  - va_list: tip de date dedicat manipulării listelor cu număr variabil de parametri (de obicei e unsigned char*)
  - va_start(va_list lp, numeArgument) : extrage în lista lp parametrii funcției care urmează după ultimul parametru fix specificat de numeArgument;
  - va_arg(va_list lp, tip_de_date): extrage la fiecare apel câte o valoare din lista lp – valoarea se consideră de tipul tip_de_date indicat ca parametru (poate fi int sau double);
  - va_end(va_list lp): obligatoriu cand se incheie operatiile pe lista lp

Funcții cu număr variabil de argumente

- **exemplul 1:** funcție ce calculează suma a n numere întregi

exemplu1.c

```c
#include <stdio.h>
#include <stdarg.h>
#include <string.h>

int suma(int n,...)
{
    int i, s;
    va_list listaParametri;
    va_start(listaParametri,n);
    s = 0;
    for(i = 0; i < n; i++)
        s = s + va_arg(listaParametri, int);
    va_end(listaParametri);
    return s;
}

int main()
{
    int a;
    a = suma(4,1,2,1,1);
    printf("a = %d\n", a);
    a = suma(5,1,2,1,1,3);
    printf("a = %d \n", a);
    return 0;
}
```

Funcții cu număr variabil de argumente

- **exemplul 1:** funcție ce calculează suma a n numere întregi

exemplu1.c

```c
#include <stdio.h>
#include <stdarg.h>
#include <string.h>

int suma(int n,...)
{
    int i, s;
    va_list listaParametri;
    va_start(listaParametri,n);
    s = 0;
    for(i = 0; i < n; i++)
        s = s + va_arg(listaParametri, int);
    va_end(listaParametri);
    return s;
}

int main()
{
    int a;
    a = suma(4,1,2,1,1);
    printf("a = %d\n", a);
    a = suma(5,1,2,1,1,3);
    printf("a = %d \n", a);
    return 0;
}
```

```
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ gcc exemplu1.c
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ ./a.out
a = 5
a = 8
```

Funcții cu număr variabil de argumente

```
void f(int x, ...)
{
 va_list lp; //declara lista de parametri
 va_start(lp ,x); //initializeaza lista de parametri, trebuie sa stiu unde incepe, dupa x
  for( ; ; ){
      tip_de_date t = va_arg(lp, tip_de_date);//extrage parametrul curent
      ... }
  va_end(lp); //elibereaza memoria
}
```

- apelam functia f dintr-o alta functie g;
- f trebuie să știe ce parametri primește;
- folosim va_start care apelează ultimul parametru formal cunoscut(x) transmis și reținut în stivă.
- transmiterea parametrilor se face de la dreapta la stânga într-o stivă

Funcții cu număr variabil de argumente

```
void f(int x, …)
{
 va_list lp; //declara lista de parametri
 va_start(lp ,x); //initializeaza lista de parametri, trebuie sa stiu unde incepe, dupa x
  for( ; ; ){
      tip_de_date t = va_arg(lp, tip_de_date);//extrage parametrul curent
      … }
  va_end(lp); //elibereaza memoria
}
```

- va_start găsește adresa lui x din stivă (e macro și nu funcție, are acces la stiva bună!) și apoi din stivă ia fiecare argument transmis cu ajutorul lui va_arg;
- va_arg trebuie să știe ce dimensiune în octeți are parametrul pe care trebuie să îl extragă din stivă;
- va_end este obligatoriu, altfel rezultatul e “undefined”;
- funcția f trebuie să știe unde se oprește cu citirea parametrilor.

Funcții cu număr variabil de argumente

- posibilă definire a macro-urilor va_list, va_start, va_arg:

```
typedef unsigned char * va_list;
#define  va_start(lp,param) (lp = (((va_list)&param) + sizeof(param)))
#define va_arg(lp,type) (*(type *)((lp += sizeof(type)) – sizeof(type)))
```

- va_list e un pointer la char (adresă de variabilă stocată pe un octet);
- va_start inițializează lista de argumente ca un pointer ce reține adresa imediată după ultimul parametru formal transmis în stivă. De la această adresa încep parametri în număr variabil;
- va_arg realizează două lucruri:
  - updateaza lista de argumente la urmatorul argument mutând pointerul (aritmetica pointerilor);
  - returneaza valoarea argumentului actual (se întoarce);

Funcții cu număr variabil de argumente

- **exemplul 1:** funcție ce calculează suma a n numere întregi

exemplu1.c

```c
#include <stdio.h>
#include <stdarg.h>
#include <string.h>

int suma(int n,...)
{
    int i, s;
    va_list listaParametri;
    va_start(listaParametri,n);
    s = 0;
    for(i = 0; i < n; i++)
        s = s + va_arg(listaParametri, int);
    va_end(listaParametri);
    return s;
}

int main()
{
    int a;
    a = suma(4,1,2,1,1);
    printf("a = %d\n", a);
    a = suma(5,1,2,1,1,3);
    printf("a = %d \n", a);
    a = suma(2,1,2,1,1,3);
    printf("a = %d \n", a);
    a = suma(7,1,2,1,1);
    printf("a = %d \n", a);
    return 0;
}
```

```
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ gcc exemplu1.c
Bogdan-Alexes-MacBook-Pro:curs11 bogdan$ ./a.out
a = 5
a = 8
a = 3
a = -515301515
```