# Programarea Calculatoarelor
# Cursul 12

Argumente din linia de comanda
Apelul funcțiilor sistem
Alte limbaje de programare

# Conținut

- Argumente din linia de comanda
  - Antetul modificat al funcţiei main
  - Conversii, sfaturi practice
  - Exemplu
- Apelul funcțiilor sistem
  - Prototipul funcției system
  - Exemplu
    - Folosirea comenzii **`dir`**
    - Folosirea compilatorului **`gcc`**
- Alte limbaje de programare

# Argumente din linia de comanda

- Se pot trimite argumente la program din linia de comanda
- Este o altă modalitate de a furniza date la program
  - Pe lângă intrarea standard (tastatura), fișiere sau direct în cod
- Folosit la aplicații de tip utilitar multifuncțional
  - Programul are comportament diferit în funcţie de argumentele de intrare
- Facilitează automatizarea şi elimină interactivitatea

# Argumente din linia de comanda

- Antetul funcției main

```
int main(int argc, char** argv)
```

sau echivalent

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

- **argc** = eng. argument count
  - numărul de argumente trimise
- **argv** = eng. argument values
  - Un tablou de șiruri de caractere
  - **argv[0]** - primul argument, este întotdeauna numele executabilului
  - **argv[1], argv[2]**, ..., **argv[argc-1]** – argumentele trimise
  - Tabloul conține și un element adițional egal cu **NULL**
    - Deci **argv[argc] = NULL**

# Argumente din linia de comanda

- Convertirea argumentelor la numere
  - Se folosesc funcții de tip atox (argument to type) din `stdlib.h`
    - `atoi, atof, atoll`, etc.
  - Se folosește funcția sscanf din `stdio.h`
    - `double x; sscanf(argv[i], "%lf", &x);`
- Argumentele sunt separate pe baza caracterului spațiu
  - Argumente care conțin spațiu trebuie încadrate între ghilimele
  - Ex: "Program Files"
- Există caractere cu rol special pentru linia de comandă
  - Trebuie introduse și tratate diferit
    - & \ < > ^ | - se prefixează cu ^, de ex. ^<
    - % - se dublează, de ex. %%
    - * ? – se pune între ghilimele simple, de ex. '*'

# Evaluarea unei expresii aritmetice - exemplu

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char** argv)
{
    if (argc < 4){
      printf("Dati 3 argumente");
      return -1;
    }

    double x, y;
    char op;
    x = atof(argv[1]);
    op = argv[2][0];
    y = atof(argv[3]);


    double rez;
    switch(op){
      case '+': rez = x+y; break;
      case '-': rez = x-y; break;
      case '*': rez = x*y; break;
      case '/': rez = x/y; break;
      case '^': rez = pow(x,y); break;
      default: rez = 0;
    }

    printf("%f", rez);
    return 0;
```

- Se trimit 3 argumente, operandul 1, operatorul și operandul 2:
- main.exe 5.1 * 6
- Interpretarea se face de către linia de comandă
- Nu merge pentru operația *
- * are rol special pentru linia de comandă și va fi înlocuit cu toate fișierele din director

# Evaluarea unei expresii aritmetice - exemplu

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char** argv)
{
    if (argc < 2){
      printf("Dati expresia incadrata intre ghilimele");
      return -1;
    }

    double x, y;
    char op;
    sscanf(argv[1], "%lf %c %lf", &x, &op, &y);


    double rez;
    switch(op){
      case '+': rez = x+y; break;
      case '-': rez = x-y; break;
      case '*': rez = x*y; break;
      case '/': rez = x/y; break;
      case '^': rez = pow(x,y); break;
      default: rez = 0;
    }

    printf("%f", rez);
    return 0;
}
```

- Se trimite 1 argument, main.exe "5.1 * 6"
- Interpretarea se face de către program
- Funcționează și *

# Redirecționare intrare/ieșire standard

- Având un executabil care citește de la intrarea standard (tastatură) și scrie pe ieșirea standard (ecran)
- În loc să se facă afișarea pe ecran se poate redirecționa afișările într-un fișier
  - se lansează executabilul folosind `main.exe > fisier_iesire`
  - dacă există fișierul acesta va fi suprascris
    - asemănător deschiderii cu modul `w`
  - pentru inserare la final, fără ștergere, se folosește `main.exe >> fisier_iesire`
- În loc să aștepte input de la tastatură se poate prelua date dintr-un fișier existent
  - se lansează executabilul folosind `main.exe < fisier_intrare`
- Se pot folosi ambele redirecționări împreună

# Apelul funcțiilor sistem

- Funcţia system din stdlib.h

```
int system(const char* command)
```

- Trimite şirul de caractere `command`, care reprezintă o comandă, la procesorul de comandă
  - Dacă se trimite NULL se verifică existenţa procesorului de comenzi
- Returnează codul returnat de comanda apelată
  - De obicei 0 înseamnă succes
  - Valori diferite de 0 semnifică diferite erori
- Interacţiune cu sistemul de operare
  - Comportament diferit pe Windows, Linux, macOs etc.
  - Este periculos – trebuie să avem drepturile necesare

# Apelul funcțiilor sistem - Afişarea fişierelor dintr-un director

- Această operaţie necesită o interacţiune cu sistemul de operare
- Se va folosi comanda DIR din Windows (`ls` pe Linux)
  - Structura comenzii

    DIR cale opţiuni
  - Cale = directorul unde se află fişierele
  - Opţiuni = argumente adiţionale care modifică comportamentul
  - Vezi DIR /? în linia de comandă pentru mai multe detalii

# Apelul funcțiilor sistem - Afişarea fişierelor dintr-un director

- Exemplu de comandă

```
DIR “d:\code\c\codeblocks\tmp\” /B > files.txt
```

- Afişează toate fişierele din directorul dat
  - În general calea poate să conţină spaţii şi este recomandat să fie încadrată între ghilimele
- Opţiunea /B (bare) afişează doar denumirea fişierelor
- Ultima parte face ca rezultatul comenzii să fie scris în fişierul **`files.txt`**, pentru prelucrare ulterioară

# Apelul funcțiilor sistem - Rularea unui program C

- Vom folosi compilatorul **`gcc`** inclus în CodeBlocks
  - Se compilează şi se linkeditează codul sursă
  - Se rulează executabilul
- Structura comenzii

```
gcc opţiuni fişier_sursă
```

  - Opţiuni = argumente adiţionale care modifică comportamentul
    - Încep cu – sau --
    - Denumirea executabilului
    - Stabilirea standardului folosit
  - Vezi gcc --help în linia de comandă pentru mai multe detalii

# Apelul funcțiilor sistem - Rularea unui program C

- Exemplu de comandă

```
gcc.exe -std=c99 -o main.exe main.c
```

  - Compilează şi linkeditează fişierul sursă main.c
  - Foloseşte standardul C99
  - Rezultatul se scrie în main.exe
  - **`gcc`** trebuie să fie vizibil
  - Se returnează 0 în caz de succes
  - **`main.c`** şi **`main.exe`** sunt în directorul curent
  - Executabilul se poate lansa cu **`system("main.exe")`**
  - Codul sursă trebuie verificat înainte de compilare și rulare din motive de securitate

# Apelul funcțiilor sistem - exemplu complex 1

```
void eval(){
  char folder[] = "d:\\code\\c\\codeblocks\\tmp2\\";

  char cmd[256];
  sprintf(cmd, "dir %s\\*.c /b > files.txt",folder);
  system(cmd);


  char fname[256];
  char gcc_root[] =
  "c:\\Program Files (x86)\\CodeBlocks\\MinGW\\bin";

  FILE* f = fopen("files.txt","r");
  FILE* fout = fopen("results.csv", "w");
  mypair p;
  int idx = 1;
  while(1){
    if (fgets(fname, 99, f) < 1)
      break;
    fname[strlen(fname)-1] = 0;
    printf("%d:\n", idx++);
    printf("processing %s\n", fname);
    remove("output.txt");
    // continuat pe slide-ul urmator
```

- funcție care evaluează corectitudinea unor programe
- folderul unde se află fișierele sursă

- listează fișierele cu extensia .c din folder și salvează lista în fișierul files.txt

- directorul unde se află compilatorul gcc

- se citește numele fișierului
- se șterge rândul nou de la finalul string-ului

- se șterge fișierul output.txt unde se vor scrie rezultatele după rularea programului

# Apelul funcțiilor sistem - exemplu complex 2

```
// ... eval
 printf("building ... ");
 sprintf(cmd, "\"%s\\gcc.exe\" -std=c99
        -o a.exe %s\\%s", gcc_root, folder, fname);

 int ret = system(cmd);
 if (ret!=0){
   p.correct = -10;
   printf("build failed\n");
 }
 else{
   printf("ok\n");
   sprintf(cmd,"a.exe");
   int ret = system(cmd);
   if (ret!=0){
     p.correct = -9;
     printf("run failed\n");
   }

   p = compare("output.txt", "chess_out_gt.txt");

 }
 printf("correct: %d / %d\n\n",
     p.correct, p.total);
 fprintf(fout, "%s, %d, %d\n",
     fname, getNum(fname), p.correct);
}
fclose(f);
fclose(fout);
}
```

- compilează și link-editează codul sursă
- se verifică dacă s-a efectuat cu succes
- se rulează executabilul
- se compară conținutul fișierului de ieșire cu cel de referință

# Alte limbaje de programare

- C++
  - Limbaj orientat pe obiecte
- Java
  - Limbaj orientat pe obiecte
- Python
  - Limbaj interpretat
- JavaScript
  - Cod rulat de browserul de web
- Matlab / FreeMat / Octave
  - Limbaj interpretat, operaţii pe matrice, un singur tip

# C++ – Hello world

```cpp
#include <iostream>

int main(){
    std::cout << "Hello, World!";
    return 0;
}
```

# C++ - numere unice dintr-un fişier

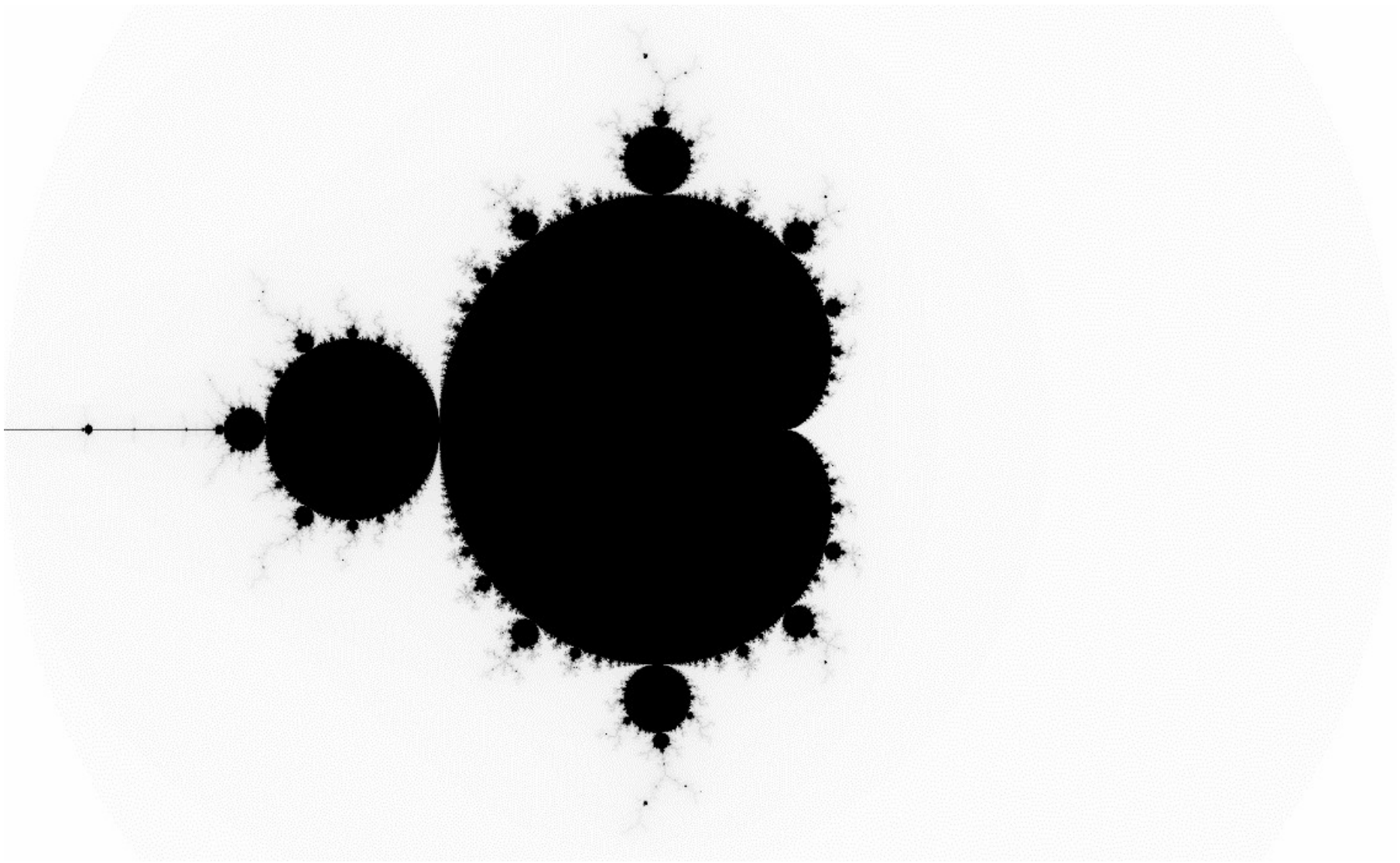
```cpp
#include <iostream>
#include <fstream>
#include <set>
using namespace std;

int main(){
  ifstream f;
  f.open("file.txt");

  set<int> S;
  while(f.peek() != EOF){
    int x;
    f >> x;
    S.insert(x);
  }
  f.close();

  for(int x : S)
    cout << x << " ";
  return 0;
}
```

# C++ cu biblioteca OpenCV - generarea fractalului Mandelbrot

```
#include <opencv2\opencv.hpp>
#include <complex>
using namespace std;
using namespace cv;
int main(){
    const int size = 2048;
    const int MAXITER = 255;
    Mat_<uchar> img(size, size);
    for (int i = 0; i < size; i++){
        for (int j = 0; j < size; j++){
            complex<double> c = { (j - size / 2.0)/size*4, (i - size / 2.0)/size*4 };
            complex<double> z = { 0, 0 };
            int iter = MAXITER;
            while (abs(z) <= 2 && --iter > 0)
                z = z*z + c;
            img(i, j) = iter;
        }
    }
    imshow("Mandelbrot", img);
    imwrite("mandelbrot.png", img);
    waitKey();
    return 0;
}
```

$$z_{n+1} = z_n^2 + c$$

# C++ - generarea fractalului Mandelbrot



# Java – Hello world

```
public class HelloWorld {
    public static void main(String[] args) {
      System.out.println("Hello, World!");
    }
}
```

# Java – Fereastră cu buton

```
import javax.swing.*;
class gui{
    public static void main(String args[]){
        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300,300);
        JButton button = new JButton("Press");
        frame.getContentPane().add(button);
        frame.setVisible(true);
    }
}
```

https://www.guru99.com/java-swing-gui.html

# JavaScript – Hello world

```
<!DOCTYPE HTML>
<html>

<body>

  <script>
    alert( 'Hello, World!' );
  </script>

</body>
</html>
```

# JavaScript – zile de naştere

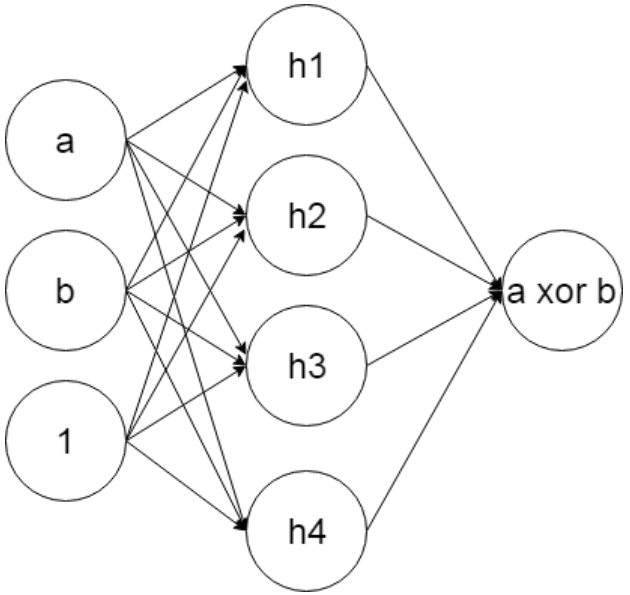
```
<!DOCTYPE html>
<html>

<script>
function f(n){
  var prob = 1;
  for(var i=0; i<n; i++)
    prob = prob * (365-i)/365;
  document.getElementById('result').innerHTML = (1-prob)*100 + '%';
}
</script>

<body>

<p> Calculate the probability that at least 2 out of n people share birthdays </p>
n = <input id="input1" />
<button onclick="f(document.getElementById('input1').value)" > Calculate </button>
<div id="result"> 0 </div>

</body>
</html>
```

# Python 3 – Hello world

```
print("Hello, World!")
```

# Python 3 – reţea neuronală xor

```
import numpy as np

X = np.array([ [0,0,1],[0,1,1],[1,0,1],[1,1,1] ])
y = np.array([[0,1,1,0]]).T
syn0 = 2*np.random.random((3,4)) - 1
syn1 = 2*np.random.random((4,1)) - 1
for j in xrange(60000):
    l1 = 1/(1+np.exp(-(np.dot(X,syn0))))
    l2 = 1/(1+np.exp(-(np.dot(l1,syn1))))
    l2_delta = (y - l2)*(l2*(1-l2))
    l1_delta = l2_delta.dot(syn1.T) * (l1 * (1-l1))
    syn1 += l1.T.dot(l2_delta)
    syn0 += X.T.dot(l1_delta)
```




https://iamtrask.github.io/2015/07/12/basic-python-network/

# Matlab – Hello world

```
‘Hello, World!’
```

# Matlab – Calcularea lui Fibonacci 1000

```
F = sym([1 1; 1 0]);
F = F^1000;
F(1,2)
```

```
434665576869374564356885276750406258025646605173717804024817290895365554179490518904038798400792551692959225930803226347752096896232398733224711616429964409065331879382989696499285160037044761377951668492288 75
```

# Semestrul 2

- Structuri de Date și Algoritmi
  - 2 grupe
- Cerc de Programare Competitiva CPC
  - Pregatire pentru ACM
  - Codeforces.com, Projecteuler.net
  - Google Kickstart, Codejam, Hashcode
  - Material mai dificil, complementar și similar cu ce se preda la SDA
    - Noțiuni elementare de C++