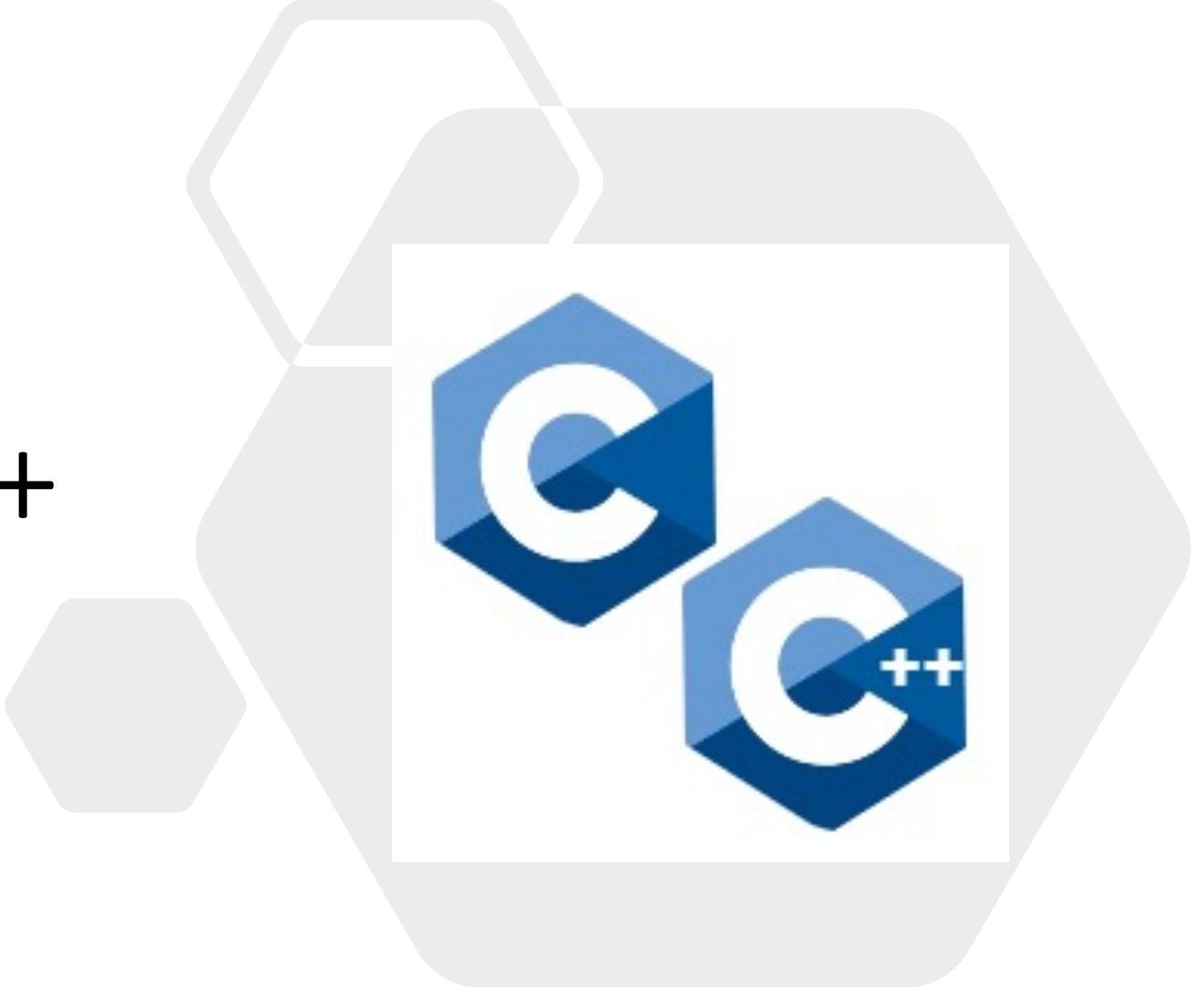


Programare II

Limbajul C/C++

CURS 1



Despre ce vom discuta?

Programare procedurală

❑ Limbajul C

- ❑ Tipuri de date și instrucțiuni
- ❑ Funcții și macro-definiții
- ❑ Tablouri și pointeri
- ❑ Structuri și uniuni

Programare orientată obiect

❑ Limbajul C++

- ❑ Concepte noi în C++
- ❑ Clase, obiecte, constructori, destructori
- ❑ Redefinirea operatorilor
- ❑ Funcții/clase şablon (template)
- ❑ Standard Template Library
- ❑ Concepte de OO
- ❑ Moștenire și polimorfism

De ce? Cât de detaliat?

Programare procedurală

Limbajul C

Dezvoltare de aplicații de **nivel scăzut**

- Aplicații pe diferite dispozitive fizice
 - Control roboți
 - Control plăci seismografice

Nivel de bază pentru cursuri *alte cursuri*

- Sisteme de operare
- Introducere în robotică
- Programarea sistemelor în timp real

Programare orientată obiect

Limbajul C++

Dezvoltare de aplicații de **nivel ridicat**

- Aplicații desktop cu interfață grafică
- Jocuri
- Aplicații Web

Nivel de bază pentru cursuri *alte cursuri*

- Programare III – concepte de OO
- Medii de programare și proiectare
- Aplicații .NET
- Securitate și criptografie

Unde este folosit?

Aplicații uzuale

- Aplicații cu interfață grafică (ex. Adobe Systems, Win Amp Media Player)
- Servere de baze de date (ex. MySQL Server, MongoDB)
- Sisteme de operare (ex. Apple OS, Microsoft Windows OS, Device-uri IoT)
- Browsere (ex. Mozilla Firefox și clientul de mail Thunderbird, Google Chrome, Safary, Opera)
- ...

[1] <https://www.codecademy.com/resources/blog/what-is-c-plus-plus-used-for/>

Domenii

- Jocuri
- Machine learning (ex. TensorFlow)
- Aplicații AR/VR
- Cercetare (ex. CERN (Organizația Europeană pentru Cercetare Nucleară), Mars Rovers)
- Software pentru avioane
- Tehnologie medicală
- ...

Bibliografie

Limbajul C

- ❑ B. Kernighan, D. Ritchie - The C Programming Language, 2nd ed., Prentice-Hall, 1988
- ❑ Ivor Horton – Beginning C: From Novice to Professional
- ❑ Steve Oualline - Practical C Programming, Third Edition
- ❑ C Programming. Brian Brown, Central Institute of Technology, NZ. Constantin quizzes
- ❑ C Programming Steven Summit, Experimental College, University of Washington, USA.
- ❑ Introduction to C Programming, University of Leicester, UK.
- ❑ C Programming. Steve Holmes, University of Strathclyde, UK.
- ❑ C Language Tutorial. Drexel University, USA. A short introduction

Limbajul C++

- ❑ Bjarne Stroustrup – The C++ Programming Language 3rd Edition. Addison Wesley, 1997.
- ❑ Kris Jamsa, Lars Klander – Totul despre C și C++. Manualul fundamental de programare în C și C++, Teora
- ❑ Andrei Alexandrescu - Programarea modernă în C++, Teora, 2002
- ❑ Octavian Catrina, Iuliana Cojocaru – Turbo C++, Teora, 1992
- ❑ N. A. Solter, S. J. Kleper - Professional C++, Wiley, 2005
- ❑ Bruce Eckel - Thinking in C++ Vol I, Vol II, 2000

Cine suntem?

Curs

Flavia Micota

Laborator

Elena Flondor

Todor Ivașcu

Emanuela Mafteiu-Scai

Flavia Micota

Valentin Pop

Toni Florea

Încurajam

Comunicarea bilaterală

La ce ne aşteptăm

Realizarea sarcinile propuse

Respectare termene

Colaborare

Sugestii pentru a putea
îmbunătăți cursul

Cum comunicăm?

- ❑ Classroom

- ❑ Materiale curs
 - ❑ Anunțuri

- ❑ Stepik

- ❑ Materiale laborator
 - ❑ Teste
 - ❑ <https://stepik.org/course/64757/syllabus>

- ❑ Script pentru note

- ❑ Ore de consultații

- ❑ Email

Cum se calculează nota finală?

Examen Final

- 30% examen scris
- 12.5% test de laborator

Evaluare pe parcurs

- 2.5% teste grilă curs platforma e-learning
- 20% test1 – nu poate fi redat
- 20% test2 – nu poate fi redat
- 2.5% punctaj obținut prin rezolvarea de probleme pe stepik
- 2.5% activitate laborator se va puncta implicarea în desfășurarea laboratorului
- 10% prezentare orală în timpul laboratorului a temelor enunțate pentru fiecare subgrupă

Ce IDE folosim?

❑ Puteți alege din o mulțime de IDE-uri disponibile

❑ Atenție

- ❑ Producător compilator C/C++
- ❑ Versiune compilator C/C++

Există diferențe în comportamentul programelor în funcție de compilatorul folosit.

Exemplu:

- [utilizator] folosirea defectuoasă a alocării dinamice
- [producător] implementare standardelor

❑ Exemple

❑ Code::Blocs

❑ Eclipse

❑ Visual Studio Code

❑ CodeLite

❑ QT Creator

❑ <https://replit.com> (online)

Cum alegem?

Orice compilator cu care vă simțiți confortabil

❑ Stepik

❑ Compilare în linia de comandă

Curs curent

- ❑ Paradigme de programare
- ❑ Particularități limbaj
 - ❑ Tipuri de date & variabile
 - ❑ Expresii & operatori
 - ❑ Instrucțiuni
 - ❑ Conditionale
 - ❑ Repetitive
 - ❑ Funcții de IO standard
 - ❑ Citirea de la tastatură
 - ❑ Scrierea pe ecran
- ❑ Bibliografie
 - ❑ Kernighan B. and D. Ritchie - The C Programming Language - capitolele 2 si 3

PARADIGME DE PROGRAMARE

- ❑ Programare nestructurată
 - ❑ Programe simple / mici ca dimensiune care conțin doar o **singură metodă**
- ❑ Programare procedurală
 - ❑ **Procedura** stochează algoritmul pe care dorim să îl (re)folosim
- ❑ Programarea modulară
 - ❑ Partiționarea programelor astfel încât datele să fie ascunse în module (**data hidding principale**)
- ❑ Abstractizarea datelor
 - ❑ Realizarea de **tipuri de date definite de utilizator** care se comportă ca și tipurile implicate
- ❑ Programarea orientată obiect
 - ❑ **Obiecte** care **interacționează**, fiecare gestionând **starea proprie**
- ❑ Programarea generică
 - ❑ Algoritmii **independenți** de **detaliile de reprezentare**

Limbajul C

- ❑ Limbaj de nivel scăzut
- ❑ Limbaj compilat
- ❑ Dezvoltat [1]
 - ❑ La începutul anii 70 de D. Ritchie (Bell Labs)
 - ❑ pentru scrierea de sistemelor de sisteme de operare (sistemul de operare UNIX pentru minicalculatoarele PDP-7 și PDP-11)
- ❑ Folosit în prezent pe scară largă
- ❑ Primul standard a fost definit în 1989 (ANSI C – 1989)
- ❑ Caracteristici
 - ❑ **Portabil**, dar **nu** independent de platformă
 - ❑ Rapid
 - ❑ Cod compact
 - ❑ Flexibil
 - ❑ Dimensiune redusă
 - ❑ ...

[1] <https://www.britannica.com/technology/C-computer-programming-language>

Pași în dezvoltarea unei aplicații

1. Scrierea programului sursă

- ❑ Fișiere text cu extensia .c și .h

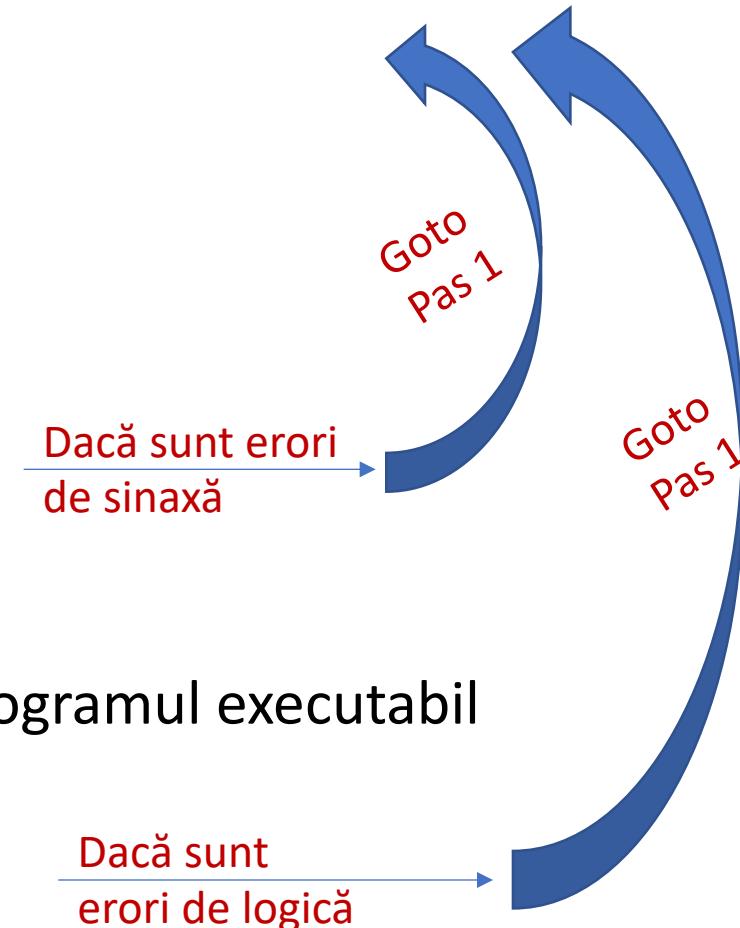
2. Compilarea

- ❑ Fișiere care conțin codul obiect al fișierului

3. Link editarea

- ❑ Legarea codului obiect și bibliotecilor -> programul executabil

4. Executarea programului + date de test



[Scrierea] Exemplu program C

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

```
int main() {  
    printf ("Hello world");  
    return 0;  
}
```

Punctul de intrare în program

Valoare de return a funcție main()

0 de obicei înseamnă că execuția s-a terminat cu succes

Directivă de preprocessare

Adăugarea bibliotecii stdio la linkeditare

Sirurile de caractere se specifică între ghilimele (ex. "Curs p2")
Caracterele se specifică între apostrofuri (ex. 'A')

Afișarea unui mesaj la consolă

La sfârșitul fiecărei linii de cod se adaugă caracterul punct și virgula (;

Pași în dezvoltarea unei aplicații

1. Scrierea programului sursă

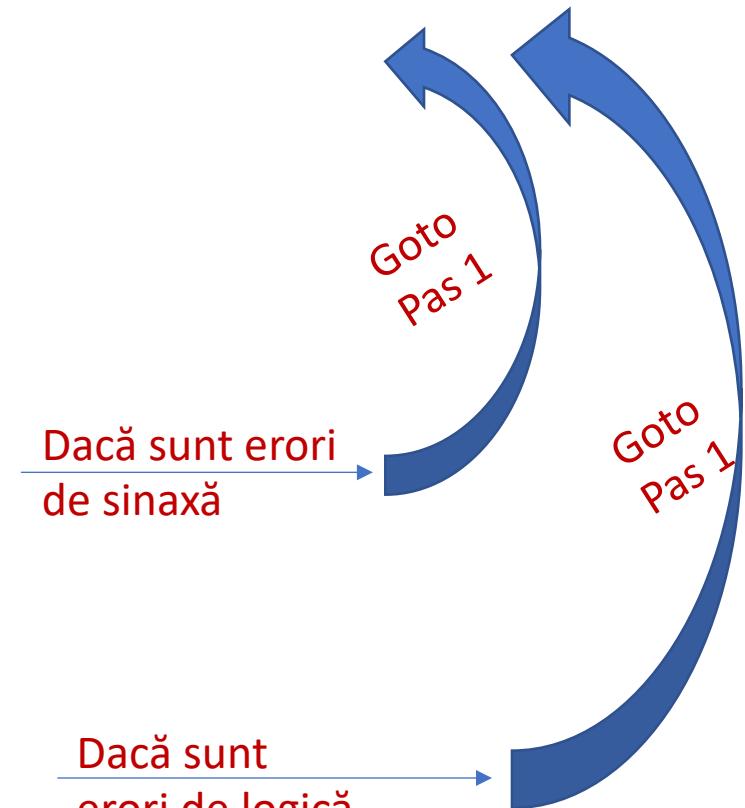
- ❑ Exemplul de mai sus îl salvăm în fișierul ex1.c

2. Compilarea & Link editarea

- ❑ [compilator] [NumeProgram].c -o NumeProgram
- ❑ Ex: gcc ex1.c -o exemplu1

3. Executarea programului + date de test

- ❑ Ex: ./exemplu1
 - ❑ În urma execuției programului pe ecran va apărea mesajul „Hello world”



Ce conține fișierul sursă?

- ❑ Un program C conține
 - ❑ Declarații de funcții și variabile globale
 - ❑ Directive de preprocessare
 - ❑ O mulțime de funcții
- ❑ Mulțime de funcții
 - ❑ Cod duplicat (**Don't repeat your self** – principiu de programare)
 - ❑ Împărțirea codului în funcții -> facilitarea înțelegерii codului
- ❑ Împărțirea codului
 - ❑ Mai multe fișiere sursă grupează funcții similare
 - ❑ De ce?
 - ❑ Nu este practic folosirea unei singure funcții de dimensiuni mari
 - ❑ Este greu de înțeles și depanat

Simțiți nevoia de a adăugă un comentariu în interiorul unei funcții pentru a explica ce urmează să se întâiple -> [refactoring] extrageți codul într-o funcție

Tipuri de date & Variabile

❑ Prin ce se caracterizează un tip de date?

- ❑ Un **domeniu** de valori
- ❑ O mulțime de **operații**
- ❑ Reprezentare internă

❑ Ce este o variabilă?

- ❑ O **asociere**/link între un **nume** și o **valoare**

❑ Cum se formează numele unei variabile?

- ❑ Poate începe cu `_` sau literă
- ❑ Contine litere și cifre

Tipuri de date

Tipuri	Nume	Reprezentare	Dimensiune	<code>unsigned</code>	<code>signed</code>	Exemple
Intregi	char	Complement față de 2	8 bits	[0, 255] [0, 2^8-1]	[-128, 127] [- 2^7 , 2^7-1]	'c' Caractere escape: '\\' '\n' '\t' '\r'
	short int		16 bits	[0, 65535] [0, $2^{16}-1$]	[-32768, 32767] [- 2^{15} , $2^{15}-1$]	123, -123 123u, 123U
	long int		32 bits	[0, 4294967295] [0, $2^{32}-1$]	[-2147483648, 2147483648] [- 2^{31} , $2^{31}-1$]	345, -345 345l, 345L
Reale	float	Virgulă flotantă	32 bits	[1.1754* 10^{-38} , 3.4028* 10^{38}]		12.4f, -12.4F
	double		64 bits	...		12.4, .5, 12., 1.4E2, 1.2e+2, 1.2E-2, 1.2e-2
	long double		80 bits	...		12.4l, -12.5L

Variabile

❑ Caracteristici

❑ Nume

❑ Valoare

❑ Clasa de memorare

❑ Durată de viață

Variabile

❑ Definire

❑ tip numeVariabilă;

❑ Exemplu

❑ int a;

❑ double a, b;

❑ Definire și inițializare

❑ tip numeVariabilă=valoare;

❑ Exemplu

❑ int a=0;

❑ int b, c=9, d=9;

Expresii & Operatori

❑ Ce este o expresie?

- ❑ O expresie conține unul sau mai mulți operatori combinați

❑ `int a = (int) (3*5.6)`

❑ Tipuri de operatori

❑ Aritmetici: +, -, %, /, *

❑ Logici: &&, ||, !

❑ Biți: <<, >>, &, |, ^

❑ Condiționali: <, >, <=, >=, ==, !=

❑ Ternar: ?: _____

❑ Virgulă: ,

❑ Prescurtați: +=, -=, *=, /=, %=, &=, |=, ^=, ++, --

❑ sizeof

❑ Asignare/atribuire: =

Ordinea de execuție:
- [P1] Constanta întreagă 3 se convertește la valoarea double 3.0 deoarece ambele operanzi trebuie să aibă același tip
- [P2] se calculează $3.0 * 5.6$
- [P3] Se convertește la tipul int rezultatul
- [P4] Se atribuie variabilei, a, valoarea

int a = x>y ? x : y;

Conversii de tip

❑ Implicită

- ❑ De la tip mai mic la tip mai mare
- ❑ `int a = 'c';`

❑ Explicită

- ❑ De la tip mai mic la tip mai mare [optional]

`int a = (int) 'c';`

- ❑ De la tip mai mare la tip mai mic [obligatoriu]

`char d = (char) 123;`

Tipul ca care se convertește se specifică în paranteze rotunde în fața expresiei/variabilei
Sintaxă: (tip) expresie

Durata de viață. Blocuri de cod

❑ Bloc de cod

- ❑ Este format din mai multe **linii de cod grupate** între accolade (**{ }**)
- ❑ Grupează linii de cod care au sens doar împreună
- ❑ În mod uzual sunt asociate cu funcțiile și corpurile instrucțiunilor
- ❑ *[Obs]* În standardul ANSI de 89 **variabile** puteau fi declarate doar la **începutul** unui bloc de cod (funcții), în standardul ANSI 99 această restricție este **înlăturată** putând fi **declarate înainte de fi folosite**

❑ Domeniul de viață al unei variabile

- ❑ Depinde de blocul de cod unde este definită

Scopul variabilelor

- ❑ Variabilele pot fi definite în afara oricărei funcții (bloc)
 - ❑ Numite variabile **globale** sau **static**e
 - ❑ Au scop global și pot fi utilizate oriunde în program
 - ❑ Consumă spațiu de stocare pe tot parcursul de rulare al programului
 - ❑ Implicit sunt **initializate** cu **zero**
- ❑ Variabile definite într-un bloc (de exemplu, funcție):
 - ❑ Numite variabile **locale** sau **automate**
 - ❑ Pot fi accesate doar în interiorul blocului în care sunt definite
 - ❑ Sunt alocate/stocate numai pe durata execuției blocului
 - ❑ Sunt **initializate** numai dacă li se dă o valoare; altfel valoarea lor este **nedefinită**
 - ❑ Sunt alocate pe stivă, gestionarea se face automat de CPU, nu trebuie alocate/deallocate manual

Clase de stocare

- ❑ extern
 - ❑ Utile mai ales dacă folosim variabile globale partajate între mai multe fișiere
- ❑ static
 - ❑ Variabilă care își păstrează valoarea la apeluri succesive ale unei funcții
- ❑ register
 - ❑ Variabilă stocată în registri
- ❑ auto
 - ❑ Clasa implicită de memorare

```
#include<stdio.h>

void fct() {
    static int a=3;
    printf("a=%d\n", a++);
}

int main() {
    fct();
    fct();
    return 0;
}
```

Exemplu

```
1: #include <stdio.h>
2: int a;
3: unsigned char c = 'A';
4: static int d;
5: extern int alpha;
6: int main(void) {
7:     extern unsigned char b;
8:     double a = 3.4;
9:     {
10:         extern a;
11:         printf("%d %d\n", b, a+1);
12:     }
13:     return 0;
14: }
```

❑ [Q1] Care sunt valorile variabilelor de la liniile 2, 4?

❑ [Q2] Care este rolul specificatorului extern?

❑ [Q3] Care este rezultatul execuției programului?

Variabile statice
a, c, d, alpha

Variabile locale
stocate în stivă

a
b

Instructiuni condiționale

Instructiunea **if**

□ Sintaxă

```
if (expresie) blocDeInstructiuni  
[else bloc de instructiuni]
```

Instructiunea **switch**

□ Sintaxă

```
switch (expresie) {  
case expresie1: blocDeInstructiuni  
case expresie2: blocDeInstructiuni  
...  
[default: blocDeInstructiuni]}
```

}

Instructiuni condiționale. Exemplu

□ Problemă: Se citesc două numere reale să se afișeze maximul dintre ele.

Instructiuni condiționale. Exemplu

■ Problemă: Se citește o cifră de la tastatură să se afișeze în cuvinte cifra citită.

Instructiuni repetitive

Instructiunea while

❑ Sintaxă

```
while (condiție)
    bloc de instructiuni
```

Instructiunea do . . . while

❑ Sintaxă

```
do {
    bloc de instructiuni
} while (condiție);
```

❑ Echivalentul în pseudocod a structuri *repeat ... until*

❑ Condiția negată

Instructiuni repetitive. Exemplu

- **Problemă:** Se citesc de la tastatură n numere naturale pozitive să se afișeze numerele pare.

Instructiuni repetitive

Instructiunea for

Sintaxă

```
for ([expresie1]; [expresie2]; [expresie 3])  
    bloc de instructiuni
```

expresie1 – initializare variabile

expresie2 – criteriu de oprire

expresie3 – modificare contor ciclu

Instructiuni de control al executiei

break

Întrerupe execuția ciclului curent

Primul nivel de imbricare

continue

- Ignoră restul ciclului, reîncepând cu evaluarea condiției

Instructiuni repetitive. Exemplu

□ **Problemă:** Să se calculeze suma primelor n numere naturale.

Instructiuni repetitive. Exemplu

□ **Problemă:** Se citesc de la tastatură n numere naturale pozitive să se afișeze numerele care au suma cifrelor cel puțin m , unde m este un număr întreg citit de la tastatură.

Instruția goto

- ❑ Instruția goto nu este **niciodată** necesară
- ❑ Codul devine greu de întreținut și înțeles
- ❑ Exceptie:
 - ❑ Dacă doriți să ieșiți din mai multe instruții repetitive imbricate pentru a raporta o eroare

```
if (critical_problem) goto error;

error: fix problem, or abort
```

Functii IO. Afisare informatii pe ecran

❑ Funcția printf()

❑ Sintaxă

❑ `printf(specificatorDeFormat [, listaDeVariabile])`

❑ unde:

❑ `specificatorDeFormat` – este un sir de caractere care contine constante si specifatori de format (specifatori de format sunt succesiuni de caractere care au semnificație pentru compilator)

- ❑ %s – pentru afisarea unei variabile de tip sir de caractere
- ❑ %d – pentru afisarea unei variabile de tip intreg
- ❑ %f – pentru afisarea unei variabile de tip float/double
- ❑ %c – pentru afisarea unei variabile de tip caracter

❑ Bibliografie

❑ Kernighan B. and D. Ritchie - The C Programming Language - capitolul 7

Functii IO. Citirea datelor de la tastatură

❑ Funcția scanf ()

❑ Sintaxă

❑ `scanf(specificatorDeFormat, listaDeVariabile)`

❑ unde:

❑ `specificatorDeFormat` – este un sir de specificatori de format (în afară de specificatori de format nu se specifică altceva, introducerea de alte caracter poate duce la comportament neașteptat)

❑ `%s` – pentru citirea unei variabile de tip sir de caractere

❑ `%d` – pentru citirea unei variabile de tip întreg

❑ `%f` – pentru citirea unei variabile de tip float/double (`%lf`)

❑ `%c` – pentru citirea unei variabile de tip caracter

Functii IO. Exemplu

- **Problemă:** Să se citească trei numere de la tastatură și să se afișeze suma și produsul lor.

Exercițiu

❑ Care este rezultatul execuției următorului cod?

```
char i,j; for(i=0;i<10,j<5;i++,j++) ;
```

Despre ce am discutat ...

Go to **www.menti.com** and use the code **8275 1199**

<https://www.menti.com/tcok5xurc1>

Despre ce vom discuta?

- ❑ Funcții

- ❑ Prototip

- ❑ Definire

- ❑ Convenția de apel

- ❑ Macro-definiții

- ❑ Bibliografie

- ❑ Kernighan B. and D. Ritchie - The C Programming Language - capitolul 4



ÎNTREBĂRI