ACADEMIA DE STUDII ECONOMICE DIN BUCUREȘTI
FACULTATEA DE CIBERNETICĂ, STATISTICĂ ȘI INFORMATICĂ ECONOMICĂ

PROGRAMARE ORIENTATĂ OBIECT

Conf. univ. dr. Bogdan IANCU

AVERTIZARE

Scopul acestui document este de a structura și de a prezenta pe scurt noțiunile discutate la curs.

Învățarea materiei exclusiv pe baza acestui material reprezintă o abordare superficială.

SĂ NE CUNOAȘTEM

Bogdan IANCU

- Programez din 2003, predau din 2012
- Am lucrat peste 10 ani ca programator în diverse companii multinaționale, acum freelancer
- Trainer acreditat de peste 10 ani
- **Email: bogdan.iancu@ie.ase.ro**

OBIECTIVE

Obiectiv principal: Prezentarea noțiunilor aferente paradigmei Orientate Obiect în limbajul C++

Obiective secundare:

- Modelarea orientată obiect
- Editare și compilare cod sursă C++
- Depanare și rezolvare erori de compilare și de execuție

EVALUARE

30% TESTE

20% test la seminar în săptămâna 7-8
10% test grilă în ultima săptămână din semestru

60% EXAMEN FINAL

Examen oral cu bilet extras și rezolvat pe calculator

10% ACTIVITATE

5% participare activă la seminar (la alegerea cadrului didactic)
5% teme de seminar (gestionate la curs)

Atenție! Obținerea a minim 1,5 puncte din cele 4 aferente activităților de pe parcurs reprezintă precondiție de intrare în examenul din sesiunea normală. Nota minimă necesară din examenul final pentru a putea promova este 5.

BIBLIOGRAFIE

- acs.ase.ro/cpp
- Ion Smeureanu – “Programarea în limbajul C/C++”, Editura CISON, 2001
- Ion Smeureanu, Marian Dardala – “Programarea orientată obiect în limbajul C++”, Editura CISON, 2002
- Bjarne Strastroup – The Creator of C++, “The C++ Programming Language”-4th Edition, Addison-Wesley
- https://cppinstitute.org/e-learning
- Alte tutoriale disponibile online

REGULI DE URMAT

E OK SĂ GREȘIM

Programele perfecte există doar în tutoriale. Programatorii profesioniști fac greșeli pe care tot ei le corectează

SUNTEM CORECȚI

Atât cu noi înșine, cât și cu ceilalți. Copierea la orice activitate (teme, teste, proiect, examen) vă poate exmatricula și vă va anula toate punctele de laborator

NU NE PLÂNGEM PÂNĂ NU ÎNCERCĂM

Totul pare greu la început. Mai întâi încercăm și, după ce irosim toate resursele, cerem ajutorul sau facem o pauză.

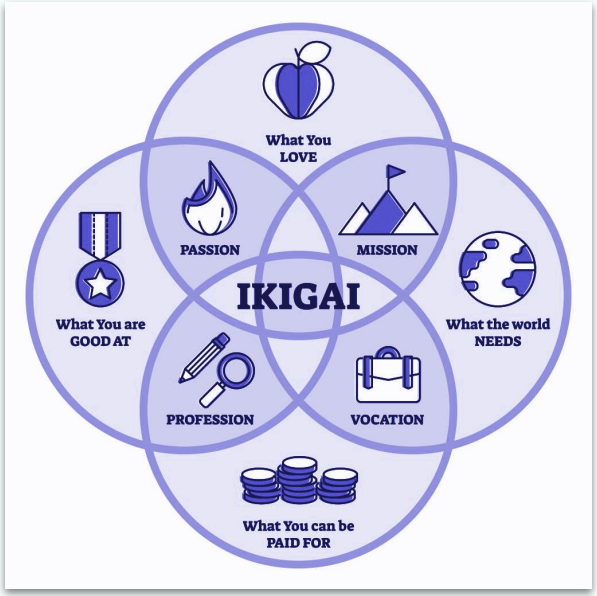
NE COMPARĂM CU NOI ÎNȘINE

Important e ca la sfârșitul semestrului să fim mai buni decât la început.

THE GROWTH MINDSET (VS. SINDROMUL VICTIMEI)

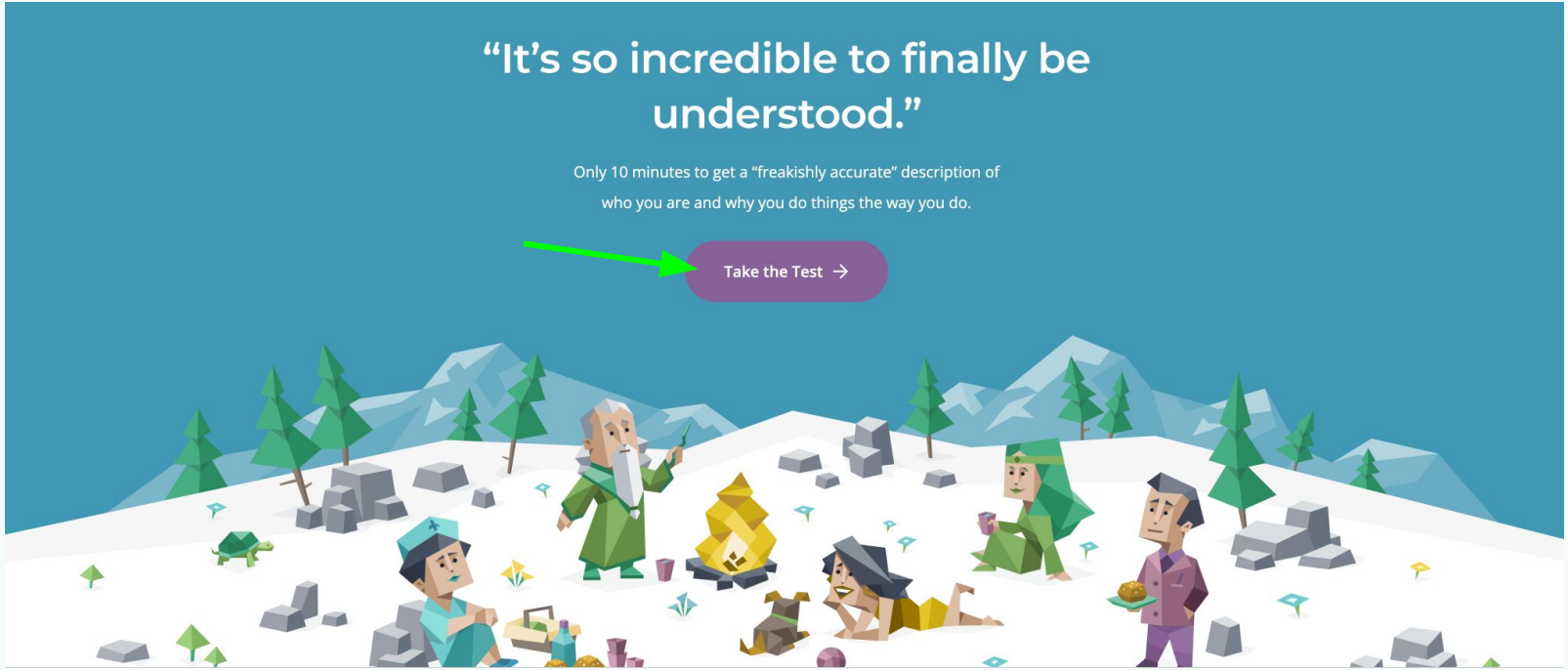
Sursă imagine:
https://sites.dartmouth.edu/learning/2017/05/18/understanding-the-growth-mindset/

PROGRAMAREA TREBUIE FĂCUTĂ CU PLĂCERE



Sursă imagine:
https://safety4sea.com/cm-ikigai-find-your-meaning-at-work-and-life/

PERSONALITATEA NOASTRĂ ARE UN IMPACT ASUPRA MODULUI ÎN CARE ÎNVĂȚĂM



10 MODALITĂȚI DE A REUȘI

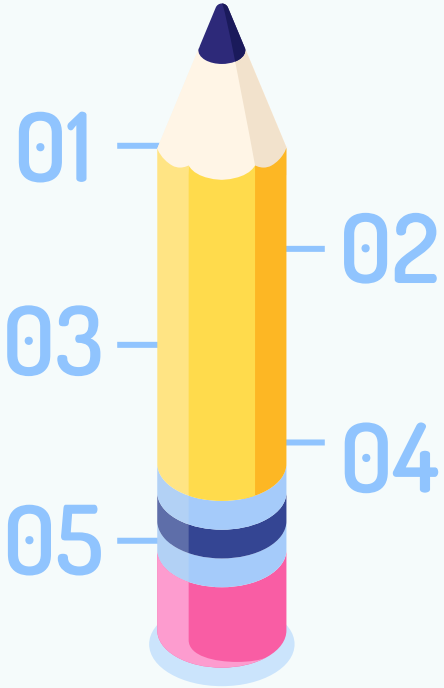
1. Scriem suficiente programe
2. Scriem suficiente programe
3. Scriem suficiente programe
4. Învățăm să folosim depanatorul
5. Punem suficiente întrebări la curs și la seminar
6. Dacă programele nu merg din prima încercare, suntem pe drumul cel bun
7. Învățăm logica din spate, nu programele pe de rost
8. Ne întrebăm „de ce?” de cel puțin 5 ori
9. Participăm activ la curs/seminar și ne facem temele
10. Cerem ajutorul celorlalți colegi și al profesorilor când suntem în impas

CE VOM ÎNVĂȚA

RECAPITULARE
Structuri, pointeri, masive, transmiterea parametrilor

FIȘIERE
Lucrul cu stream-uri (fișiere standard, text și binare)

STL
Clase template, Standard Template Library



CLASE
Definire, atribute, metode, constructori, operatori

DERIVARE
Ierarhii de clase, polimorfism, funcții virtuale

Întrebare

Ce limbaj ați studiat în liceu și/sau în anul 1?

Dezbatere

De ce P.O.O.?

Care este motivul pentru care paradigma clasică (structurată) a trebuit înlocuită?

PATRU MOTIVE PRINCIPALE

ABSTRACTIZARE

Elementele din lumea reală pot fi modelate mai ușor în cod

POLIMORFISM

Aceeași entitate poate avea mai multe comportamente în funcție de context

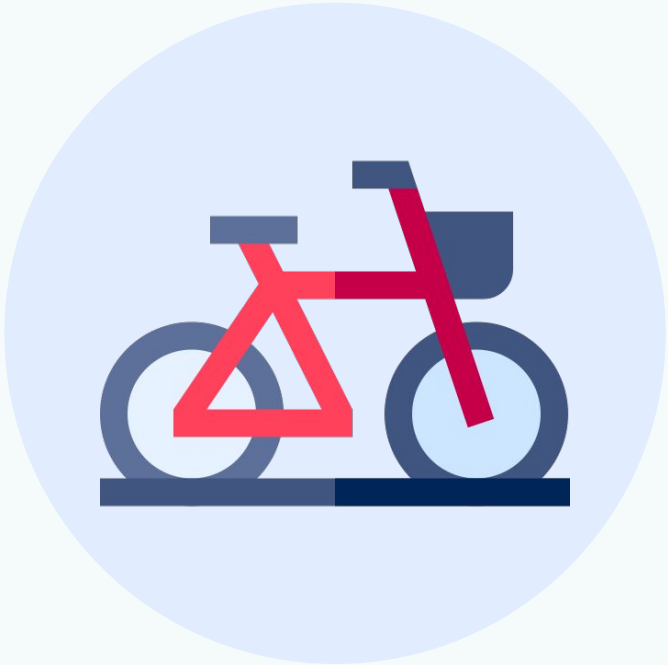
ÎNCAPSULARE

Entitățile pot ascunde informații sau comportamente de alte entități

DERIVARE

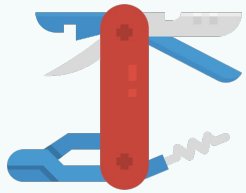
Se pot crea entități noi pe baza entităților existente

C vs C++



vs

De ce C++?



- Foarte rapid
- În top 5 limbaje la nivel mondial de peste 30 de ani (https://www.tiobe.com/tiobe-index/)
- Încorporează toate conceptele de Programare Orientată Obiect într-un mod pur
- Are sintaxa bazată pe cea a limbajului C
- Cuțitul elvețian (îmbunătățit) al limbajelor de programare

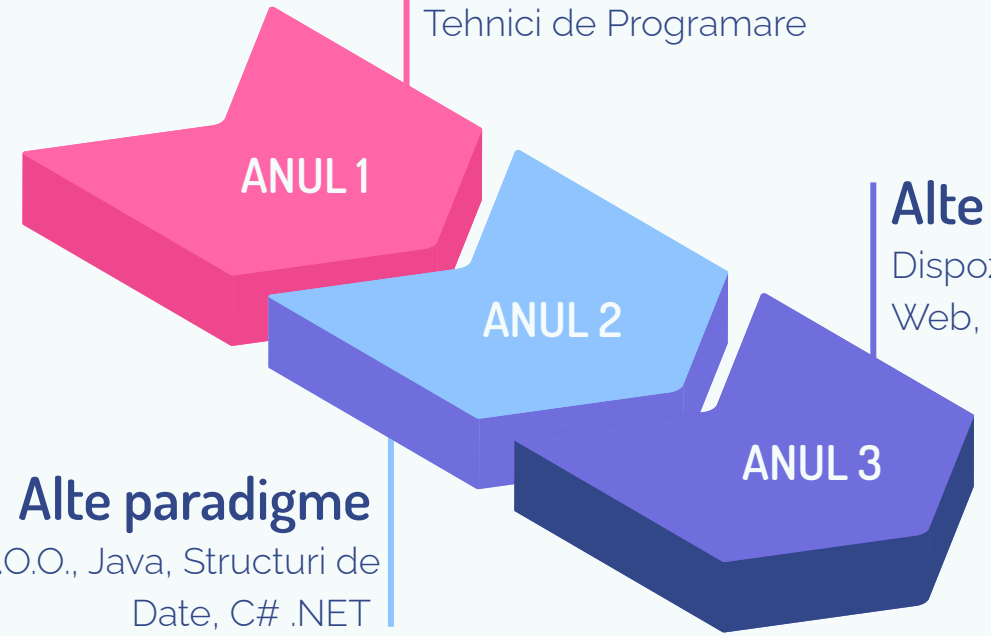
Dennis Ritchie (1941 – 2011) - creatorul limbajului C

„Singura modalitate de a învăța un limbaj nou de programare este prin a scrie programe în acel limbaj”

DRUMUL CĂTRE UN PROGRAMATOR COMPLET

Programare structurată

Bazele Programării, Algoritmi și Tehnici de Programare



Alte tipuri de aplicații + calitate

Dispozitive și Aplicații Mobile, Tehnologii Web, Calitate și Testare Software

Alte paradigme

P.O.O., Java, Structuri de Date, C# .NET

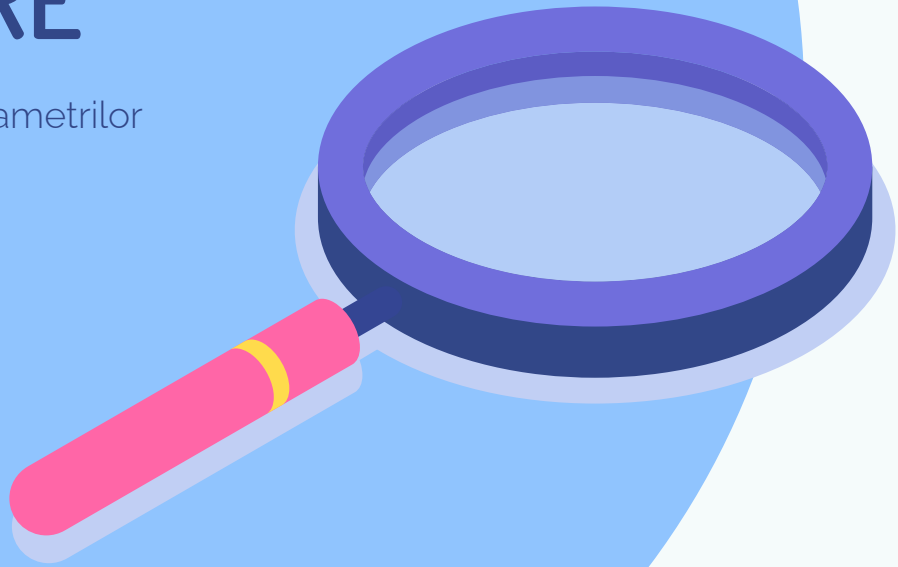
Ce medii de dezvoltare vom folosi?

- Microsoft Visual Studio 2022 Community la curs și laborator (https://visualstudio.microsoft.com/downloads/)
- Orice compilator și editor (chiar și online) pentru a învăța pe cont propriu
- **Atenție!** Chiar dacă paradigma este aceeași, diferite compilatoare de C++ pot returna rezultate diferite pentru același cod sursă

Curs 01

RECAPITULARE

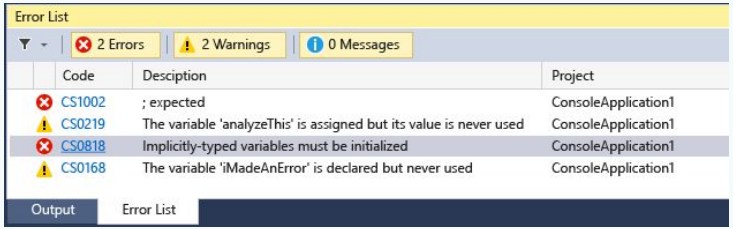
Pointeri, masive, transmiterea parametrilor



Întrebare

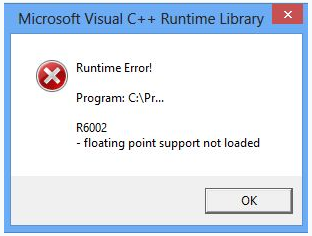
Câte tipuri de erori întâlnim în C++?

Erori de compilare



- Generate de editarea greșită a codului
- Cauza principală este dată de faptul că nu se cunoaște sintaxa
- Sunt ușor de corectat dacă se citește cu atenție mesajul de eroare returnat de mediul de dezvoltare
- E indicat să folosim opțiunea Build sau Rebuild după fiecare corecție pentru a vedea dacă eroarea persistă
- A nu se confunda cu avertizările (warnings) ce lasă programul să ruleze, dar indică posibile probleme ce pot apărea în diferite situații

Erori de execuție

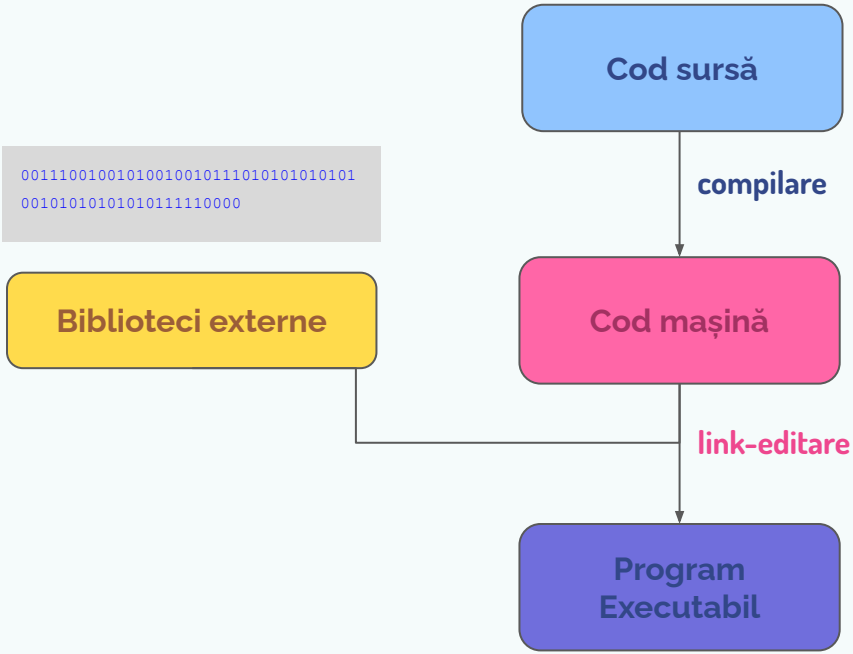


- Generate de greșeli de logică (împărțiri la zero, dezalocarea unei zone de memorie deja dezalocată, accesarea elemntelor unui vector după dimensiunea sa maximă, etc.)
- În C++ majoritatea sunt cauzate de înțelegerea superficială a gestionării memoriei (sau învățarea pe de rost a anumitor algoritmi)
- Pentru a putea fi corectate e nevoie să folosim depanatorul (debugger-ul) pentru a vedea în timp real ce valori există în variabile

Întrebare

Care sunt etapele prin care trece un program de la scriere până la execuție?

Drumul de la cod sursă la execuție




```
#include <stdio.h>

int main() {
 printf("Hello, World!");
 return 0;
}
```

```
00111001001010010010111010101010101
00101010101010111110000
```

```
11100011000001100001100010101010010
10010010111000010010010010100100010
10001001001001001
```

```
11000101010100101011100101000101010
10101001010101110001010101010001010
10101001000001100010101010010101010
10101110000
```

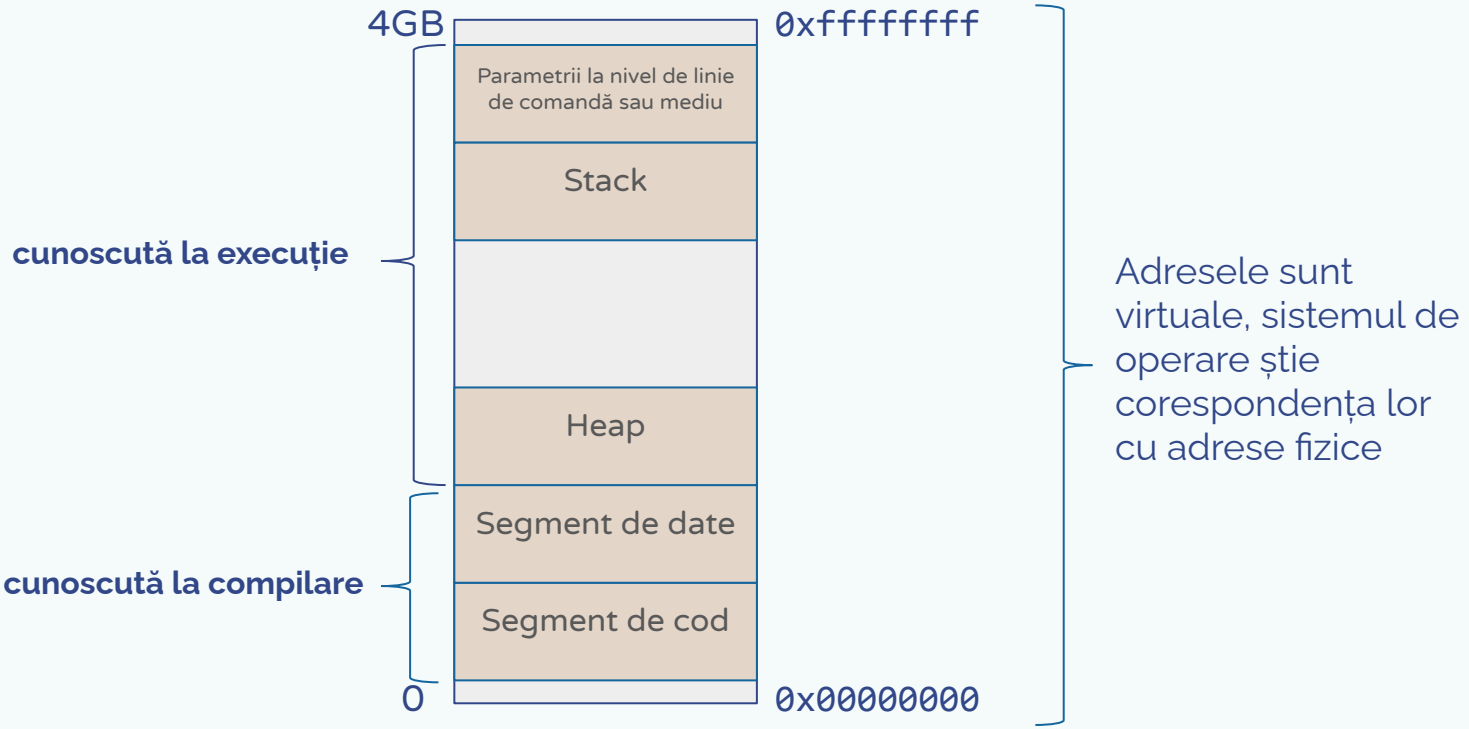
Întrebare

Ce este un pointer?

Întrebare

Câte tipuri de memorie folosește limbajul C/C++?

Cum arată memoria



Întrebare

Unde sunt salvați pointerii?