Semelbauer Ștefan

INTRODUCERE

Calculatoarele sunt unelte foarte puternice care pot efectua operații matematice mult mai rapid decât un om. Toți le folosim, fie că este vorba despre laptopul pe care ne uităm la seriale, smartphone-ul cu care încărcăm selfie-uri pe rețelele sociale sau serverul pe care Facebook stochează pozele pe care le încărcăm.

Cu toată viteza și capacitatea imensă de stocare a calculatoarelor, ele nu au o capacitate de înțelegere asemănătoare cu a oamenilor, deci fiecare comandă pe care vrem să le-o transmitem trebuie formulată într-un limbaj foarte clar care să poată fi ușor înțeles de către ele.

Calculatoarele sunt atât de proaste încât chiar dacă uităm o virgulă sau o literă, nu mai înțeleg comanda și dau o eroare.

Dacă pentru un om este foarte ușor să citească și să înțeleagă *opropoziţiescrisăfărăspaţii*, pentru un calculator lucrurile nu sunt la fel de ușoare.

Din această cauză ne și lovim deseori de programe care dau erori. Fiecare din noi și-a introdus greșit cel puțin o dată adresa de email și a primit o eroare.

Cum funcționează calculatoarele

Pentru a da comenzi unui calculator este nevoie să folosim un limbaj de programare. Putem să ne imaginăm limbajul de programare ca și o limbă pe care o folosim ca să comunicăm cu calculatorul, iar programele pe care le scriem ca pe niște scrisori pe care i le trimitem calculatorului. La fel cum scrisorile sunt formate din mai multe cuvinte, așa și programele sunt formate din mai multe instrucțiuni. Mai mult, la fel cum citim cuvintele de la stânga la dreapta, de sus în jos, și calculatorul citește instrucțiunile de la stânga la dreapta și de sus în jos.

Fiecare instrucțiune îi spune calculatorului ce să facă. Putem să ne imaginăm instrucțiunile date calculatorului în felul următor:

Calculează 2+3

Afișează pe ecran "bine ai venit!"

Pentru a realiza operații mai complexe, avem nevoie de mai multe instrucțiuni. Mai mult, fiecare instrucțiune poate folosi rezultatele instrucțiunilor executate anterior. Ne putem imagina acest lucru în felul următor:

Găsește poza utilizatorului Andrei
Afișează poza găsită

CUM SA FOLOSESTI PLATFORMA

De multe ori programarea poate părea grea sau chiar imposibilă. E normal să te simți din când în când descurajat și să vrei să renunți. O să vezi că pe măsură ce lucrezi și treci peste obstacole vei deveni un programator bun.

Nu uita că fiecare programator de top a fost fix în punctul în care te afli tu acum.

C++

Înainte să treci mai departe, te rog aruncă un ochi peste secțiunea cu întrebări frecvente care poate fi accesată dând click AICI

Limbajul de programare folosit în acest curs este C++.

C++ este folosit în majoritatea liceelor din România și este limbajul cerut la multe dintre facultățile de profil în cadrul admiterii. Limbajele C și C++ ocupă locurile 2 și 3 în topul limbajelor de programare realizat de tiobe. https://www.tiobe.com/tiobe-index/

Un program în C++ care afișează pe ecran textul "Salut" arată în felul următor:

```
#include <iostream>
using namespace std;

int main() {
        cout<<"Salut";
        return 0;
}
```

Conținutul unui program se numește cod sursă. Pentru a nu trebui să scriem la tastatură de fiecare dată programul, putem să salvăm codul sursă al programului ca un fișier text.

Ce se întâmplă în spate

Totuși, calculatorul nu înțelege decât limbajul mașină, care este un limbaj format doar din cifrele `0` și `1`. Limbajele de programare mai avansate au fost inventate pentru a ne ușura munca și pentru a nu ne chinui să scriem programe formate doar din `0` și `1`.

Din acest motiv avem nevoie de un program care să traducă programul scris de noi în C++ într-un program în limbaj mașină care să poată fi înțeles și executat de către calculator. Programele care fac traducerea se numesc compilatoare.

Compilatorul citește codul sursă al programului și generează un fișier executabil care conține codul mașină și care poate fi executat de către calculator. Fiecare program pe care îl pornim pe calculator este un fișier executabil. Odată generat, fișierul executabil poate fi pornit de oricâte ori, iar calculatorul va executa instrucțiunile din el.

Atunci când programul nu respectă regulile de limbaj, compilatorul nu va reuși să traducă programul tău în cod mașină și te va avertiza printr-o *eroare de compilare.*

Chiar dacă sună complicat, în realitate nu trebuie să îți bați capul cu toate detaliile acestea. După ce ai scris programul, vei apăsa un singur buton și calculatorul va încerca să îți ruleze programul.

Mai jos este un exemplu de program greșit și eroarea care apare la compilare.

```
#include <iostream>

int main() {
        cout<<"Salut";
        return 0;
}
```

Eroare:

```
prog.cpp: In function ‘int main()’:
prog.cpp:4:2: error: ‘cout’ was not declared in this scope
```

Poți să încerci să modifici programul de mai jos și să testezi ce se întâmplă. Pentru a rula programul, apasă pe triunghiul gri din dreapta jos.

COMPILATOR ONLINE

HELLO WORLD

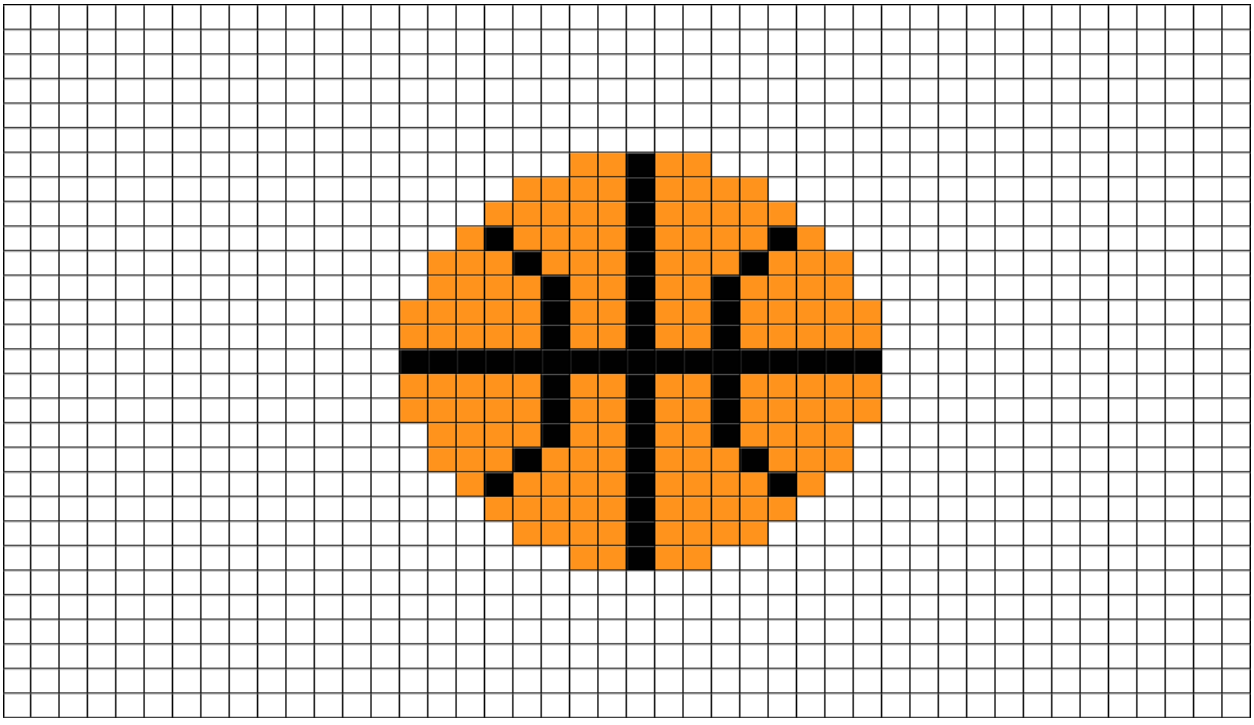
După cum am văzut în lecția anterioară, un program care afișează pe ecran textul Hello world este destul de scurt. Totuși spre deosebire de limbajul natural, programul conține multe instrucțiuni pe care probabil nu le înțelegi. Pe parcursul lecției vom lua fiecare linie și vom explica ce face.

Întrucât ești la început, nu trebuie să îți faci griji dacă nu ai înțeles exact cum funcționează sau ce rol au instrucțiunile prezentate mai jos. Pe măsură ce înțelegi conceptele prezentate pe parcursul cursului, vei reuși să înțelegi în detaliu aceste instrucțiuni și rolul lor.

```
#include <iostream>
```

În general rezultatele rulării unui program sunt afișate pe un ecran. Fie că este vorba despre ecranul smartphone-ului, cel al laptopului sau e vorba de monitorul de la bancomat, avem nevoie de o reprezentare vizuală a rezultatelor.

Un monitor are mulți pixeli care sunt așezați sub forma unui dreptunghi și care iau diverse culori pentru a afișa o imagine. Dacă monitoarele ar avea foarte puțini pixeli, imaginile afișate de ele ar arăta în felul următor



Întrucât singurul lucru pe care monitorul știe să îl facă e să coloreze pixelii în diverse culori, calculatorul trebuie să îi spună care dintre pixeli trebuie să îi coloreze și ce culori să folosească.

Ar fi o treabă foarte grea dacă ar trebui să îi spunem calculatorului pe care pixeli trebuie să îi coloreze pentru a afișa litera H, pe care să îi coloreze pentru litera e pentru a apărea imediat după H și așa mai departe.

Pentru a nu fi nevoiți să ne batem capul legat de afișarea textului pe ecran, putem folosi programe scrise de alții care să ne ajute. Scriind `#include <iostream>`, îi spunem compilatorului că dorim să folosim programe din biblioteca iostream. Iostream (de la **i**nput **o**utput **stream**) este o colecție de programe care ne ajută la operațiile de citire de la tastatură și scriere pe ecran.

În cadrul unui program putem include mai multe biblioteci, dar fiecare instrucțiune de includere trebuie să fie pe o linie separată. De exemplu:

```
#include <iostream>
#include <fstream>
#include <algorithm>
```

```
using namespace std;
```

Un namespace este un dicționar care ne ajută să identificăm și să folosim diferite componente ale unui program. `using namespace std;` indică compilatorului că dorim să folosim namespace-ul standard pentru C++.

Pentru a înțelege noțiunea, putem face următoarea analogie cu o companie de taxi. În cazul existenței unei singure companii *ABC Taxi*, căutarea numărului de telefon ar fi simplă datorită faptului că nu poate fi confundată cu alte companii de taxi. În cazul mai multor companii de taxi cu același nume, trebuie precizat, spre exemplu, orașul în care se află compania. Astfel, poți căuta *ABC taxi* în Cluj, pentru că există o singură astfel de companie. In acest mod, Cluj devine namespace-ul lui *ABC taxi*.

Practic `using namespace std;` te ajută să folosești anumite programe din bibliotecile limbajului pe baza numelui lor.

```
int main()
```

De multe ori programele conțin multe instrucțiuni. Întrucât calculatorul nu știe să se uite la o listă de instrucțiuni și să decidă de unde să înceapă, el are nevoie să îi spunem acest lucru.

`int main()` este numele folosit pentru a identifica locul de unde să înceapă executarea instrucțiunilor programului

`{` și `}`

Acoladele (parantezele ondulate) sunt folosite pentru a marca începutul și sfârșitul unui grup de instrucțiuni. În acest caz sunt folosite pentru a marca începutul și sfârșitul lui `int main()`.

```
cout<<"hello world";
```

`cout` este o instrucțiune din biblioteca iostream care ne ajută să afișăm diverse lucruri pe ecran. În acest caz ne ajută să afișăm textul dintre ghilimele pe ecran. Dacă schimbăm instrucțiunea în `cout<<"wellcode te invata sa programezi";` atunci la execuție pe ecran va apărea noul text.

```
return 0;
```

Pentru a putea folosi un calculator, avem nevoie să avem instalat pe el un sistem de operare. Majoritatea oamenilor folosesc windows și nu cred că este nevoie să îți explic ce este acesta.

De fiecare dată când dăm dublu click pentru a porni un program, sistemul de operare este cel care îi spune calculatorului să pornească programul. Programul poate să îi spună sistemului de operare să pornească alte programe și de aceea sistemul de operare trebuie să mențină deschisă o linie de comunicare cu programul pornit.

Atunci când programul pe care l-am pornit își termină execuția sau este închis, trebuie să îi spună sistemului de operare acest lucru pentru a închide linia de comunicare cu el.

`return 0` atunci când se află în interiorul lui int main(), îi spune sistemului de operare că execuția programului tău s-a încheiat cu succes.

; la finalul liniei

Probabil ai observat că după unele instrucțiuni am pus `;`. Caracterul `;` marchează sfârșitul unei instrucțiuni și va trebui să îl pui după fiecare instrucțiune. Dacă nu îl pui, vei avea o eroare de compilare, pe care e ușor să o identifici pentru că îți spune chiar asta: `expected ';' before ...`.

Observație: După `#include <iostream>` și `int main()` nu se pune `;`. Prima se termină atunci când se trece pe o linie nouă, iar a doua e urmată de grupul de instrucțiuni dintre `{}`.

Poți să modifici textul dintre ghilimele în programul de mai jos pentru a vedea cum se modifică textul afișat.

COMPILATOR ONLINE

EVALUAREA PROBLEMELOR

Felicitări, ai ajuns la prima problemă de pe site. Uită-te la filmulețul de mai

jos pentru a afla cum funcționează evaluarea problemelor.

Restricțiile

Deoarece programul tău va fi rulat pe mai multe date de intrare (exemple) posibile, este important să știi cât de mari pot fi datele introduse în program pentru testare. În rubrica de restricții a fiecărei probleme ți se va spune acest lucru.

De exemplu dacă ți se spune că programul tău va citi un număr, iar în rubrica de restricții se spune că numărul nu va fi niciodată mai mare decât `100`, atunci înseamnă că programului tău îi va fi dat tot timpul un număr mai mic sau egal cu `100` atunci când este testat de către noi.

Testarea în CodeBlocks

Ai grijă ca atunci când testezi pe exemplu, programul tău să afișeze **DOAR** datele de ieșire din exemplu, **EXACT** în formatul specificat. Dacă vei afișa orice alte mesaje în plus, vei lua 0 puncte.

AFISARE

Cerință

Scrie un program care afișează pe ecran mesajul `Vreau sa invat sa programez!`.

Date de intrare

Pentru această problemă nu există date de intrare.

Restricții

Întrucât nu există date de intrare, nu există nici restricții asupra lor

Date de ieșire

Pe prima linie se va afișa mesajul `Vreau sa invat sa programez!`

Exemplu

Intrare

Ieșire

```
Vreau sa invat sa programez!
```

Poți să scrii și să testezi programul tău mai jos. Pentru a-l rula, apasă pe triunghiul gri.

COMPILATOR ONLINE

Pentru a vedea dacă programul scris de tine este corect, copiază programul mai jos și apasă pe butonul de trimitere.

Trimite o soluție

CODEBLOCKS ONLINE (TESTER-ul lor).