

**PROIECT**

la Programare Orientată pe Obiecte

**Spânzurătoarea**

Student: Hreceniuc Larisa

Grupa : 3121 A

**CUPRINS**

[TEMA ŞI MOTIVAŢIA ALEGERII 3](#_Toc377654768)

[1.ELEMENTE TEORETICE 4](#_Toc377654769)

[1.1 DESCRIEREA PROBLEMEI 4](#_Toc377654770)

[1.2 ELEMENTE SPECIFICE POO 5](#_Toc377654771)

[1.3 ABORDAREA TEORETICĂ A PROBLEMEI 6](#_Toc377654772)

[2.IMPLEMENTARE 7](#_Toc377654773)

[2.1 TEHNOLOGII FOLOSITE 7](#_Toc377654774)

[2.2 DIAGRAMA DE CLASE 8](#_Toc377654775)

[3. ANALIZA SOLUŢIEI IMPLEMENTATE 10](#_Toc377654776)

[3.1 FORMATUL DATELOR DE I/O 10](#_Toc377654777)

[3.2 STUDII DE CAZ 12](#_Toc377654778)

[3.3 MANUAL DE UTILIZARE 16](#_Toc377654779)

[4.CONCLUZII 17](#_Toc377654780)

[5.BIBLIOGRAFIE 18](#_Toc377654781)

**TEMĂ PROIECT**

# TEMA ŞI MOTIVAŢIA ALEGERII

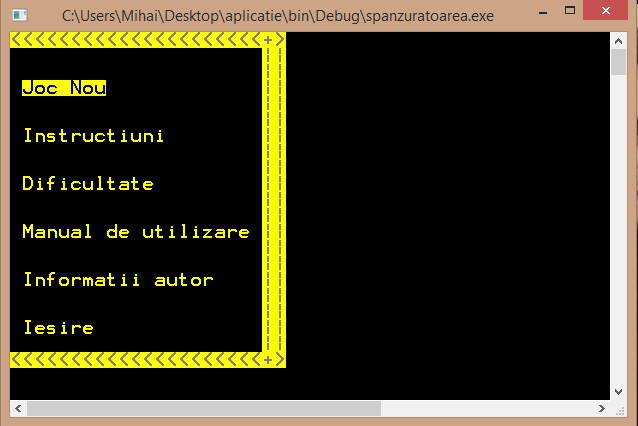
Tema acestui proiect este jocul spânzurătoarea. Realizarea unei aplicaţii ce va implementa joculeţul spânzurătoarea. Ca şi motivaţie a alegerii mele pot menţiona pasiunea pentru joculeţe şi provocarea de a realiza unul de genul de la zero în cadrul programării orientate pe obiecte şi folosind limbajul de programare C++.

**CAPITOLUL 1**

## 1.ELEMENTE TEORETICE

## 1.1 DESCRIEREA PROBLEMEI

Jocul spânzurătoarea este un joc de logică în care trebuie să ghiceşti unul sau mai multe cuvinte, prin introduceri succesive de litere. Cuvintele dispuse pe trei nivele de dificultate, sunt alese random de către program, din trei liste de cuvinte. La începerea unui joc nou, în funcţie de dificultatea aleasă, se generează un cuvânt random, mai precis lungimea acelui cuvânt marcata printr-un identificator necunoscut: ”\_”. Utilizatorul are posibilitatea să ghicească cuvântul din şase încercări. Dacă litera introdusă de utilizator este corectă (lucru verificat de functia introducereLiteră), acea literă va fi descoperită în înşiruirea identificatorului “\_” iar numarul de încercari rămase nu se va modifica. Dacă litera introdusă de utilizator este greşită, o parte dintr-un corp de om va fi adaugată la spânzurătoare iar numarul de încercari rămase va scădea cu unu şi totodata lângă spâzurătoare se va creea o listă cu literele introduse care sunt greşite, cu scopul de a nu fi repetate în încercarile viitoare. În momentul în care toate cele şase încercari au fost utilizate şi cuvântul nu a fost ghicit, corpul uman, alcatuit din şase parţi: cap, trup, doua mâini, doua picioare, va fi spânzurat, ceea ce înseamna sfârşitul jocului pentru utilizatorul respectiv; utilizator caruia i se acordă şansa de a începe un joc nou. Pentru a începe un joc nou, utilizatorii trebuie ca prima dată să selecteze un nivel de dificultate. Nivele de dificultate sunt în numar de trei: 1.usor 2.mediu 3.greu, fiecarui nivel îi corespunde o listă de cuvinte din limba română, de la cuvinte scurte la cuvinte foarte lungi. Meniul jocului mai dispune de instrucţiuni în care este prezentat pe scurt jocul şi tot pe scurt cum funcţioneaza jocul, deasemenea de un scurt manual de utilizare şi de opţiunea de ieşire din joc cu posibilitatea de întoarcere în meniu. Mai jos este dată structura meniului jocului:

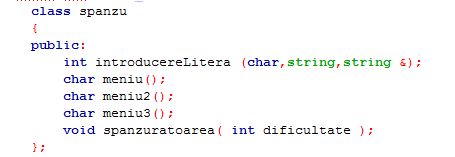


## 1.2 ELEMENTE SPECIFICE POO

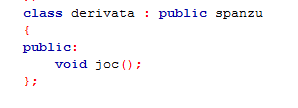
Dintre elementele specifice POO, în proiectul curent găsim:

* Clase
* Moştenirea
* Transmiterea parametrilor prin valoare

**Clasa** reprezintă un tip abstract de dată, ce încapsulează atât elementele de date(datele membre) pentru care s-a adoptat un anumit mod de reprezentare cât şi operaţiile asupra datelor (funcţiilemembre,metode). O clasă din proiect:



**Moştenirea** . În c++ întâlnim noţiunea de derivare, care este în fapt o abstractizare a noţiunii de moştenire. O clasă care adaugă proprietăţi noi la o clasă deja existentă, vom spune ca este derivată din clasa originală. Clasa originală poarta denumirea de clasă de bază. O moştenire din proiect:

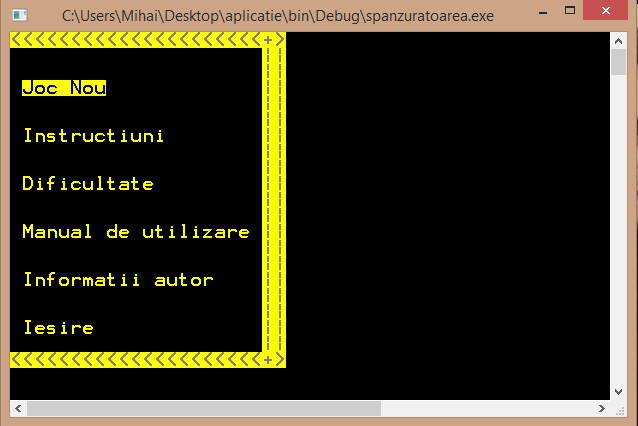


**Transmiterea parametrilor prin valoare**. De la programul apelant către funcţia apelată, prin apel, se transmit valorile partametrilor efectivi, reali. Aceste valori vor fi atribuite, la apel, parametrilor formali. Deci procedeul de transmitere a parametrilor prin valoare constă în încărcarea valorii parametrilor efectivi în zona de memorie a parametrilor formali (în stivă). La apelul unei funcţii, parametrii reali trebuie să corespundă - ca ordine şi tip - cu cei formali. Exemplu din proiectul curent:

C:\Users\andy\Desktop\Screenshot_10.png

## 1.3 ABORDAREA TEORETICĂ A PROBLEMEI

Proiectul este construit în mod consolă, iar grafica este de tip text. Interfaţa aplicaţiei este una simplă, la nivelul a ceea ce poate oferi mediul de dezvoltare Code::Blocks. Interfaţa este sub forma unui meniu, accesul la o opţiune facându-se prin selectarea optiunii dorite .



**CAPITOLUL II**

### 2.IMPLEMENTARE

Aplicaţia „Spânzurătoarea”este creată în Code::Blocks şi se bazează pe conceptul de *clasă*, specific Programarii Orientate pe Obiecte și în cadrul ei sunt utilizate tehnologiile specifice acestui tip de programare, cum ar fi: clase, moştenire, supradefinirea operatorilor, transmiterea parametrilor prin valoare.

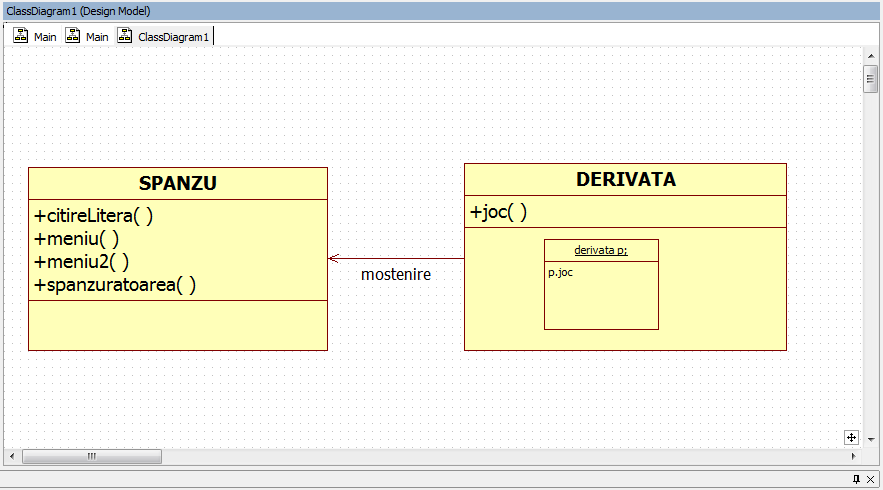
### 2.1 TEHNOLOGII FOLOSITE

**Clasa.** Clasa este asemănătoare cu o structură din limbajul C, dar care poate avea în componenţa sa membri atât de tip variabilş cât şi de tip funcţie. Pentru datele din interiorul clasei se utilizează de obicei termenul de date membre, iar pentru funcţii denumirea de funcţii membre sau metode. O clasa permite încapsularea în interiorul său a datelor si a codului.Întocmai ca în limbajul C, pentru a putea utiliza efectiv un tip de date (în cazul de fata o clasă), trebuie să definim o variabiă de acel tip. **Moştenirea.** O caracteristică importantă a claselor C++ o reprezintă conceptul de moştenire. Acest concept permite crearea unui obiect derivat dintr-un altul, ca atare noul obiect poate include membrii vechi şi, în plus, proprii săi membrii. În c++ întâlnim noţiunea de derivare, care este în fapt o abstractizare a noţiunii de moştenire. O clasă care adaugă proprietăţi noi la o clasă deja existentă, vom spune ca este derivată din clasa originală. Clasa originală poarta denumirea de clasă de bază. **Supradefinirea operatorilor**. Aşa cum operatorii pot fi folosiţi în limbajul C++ pentru tipurile de date predefinite, ei pot fi folosiţi şi pentru tipurile de date introduce de programatori. Cu toate ca nu se pot creea operatori noi, majoritatea celor existenţi pot fi supraîncărcaţi ca să poată fi folosiţi şi pentru noile clase. Acesta este unul din cele mai puternice attribute ale limbajului C++. Chiar dacă supraîncărcarea operatorilor face programul mai clar decât folosirea unor funcţii pentru aceleaşi operaţii, folosirea excesivă a acestei tehnici, poate face uneori ca aplicaţiile sa devină criptice si dificil de înţeles. **Transmiterea parametrilor prin valoare.** De la programul apelant către funcţia apelată, prin apel, se transmit valorile partametrilor efectivi, reali. Aceste valori vor fi atribuite, la apel, parametrilor formali. Deci procedeul de transmitere a parametrilor prin valoare constă în încărcarea valorii parametrilor efectivi în zona de memorie a parametrilor formali (în stivă). La apelul unei funcţii, parametrii reali trebuie să corespundă - ca ordine şi tip - cu cei formali.

**Limbajul de programare** folosit în acest proiect este **C++.** [C++](http://ro.wikipedia.org/wiki/C%2B%2B) (pronunțat în română "C plus plus" și în engleză "si plas plas") este un limbaj de programare general, compilat. Este un limbaj multi-paradigmă, cu verificarea statică a tipului variabilelor ce suportă programare procedurală, abstractizare a datelor, programare orientată pe obiecte. În anii 1990, C++ a devenit unul din cele mai populare limbaje de programare comerciale, rămânând astfel până azi. Bjarne Stroustrup de la Bell Labs a dezvoltat C++ (inițial denumit C cu clase) în anii 1980, ca o serie de îmbunătățiri ale limbajului C. Acestea au început cu adăugarea noțiunii de clase, apoi de funcții virtuale, suprascrierea operatorilor, moștenire multiplă, șabloane (engleză template) și excepții. Limbajul de programare C++ a fost standardizat în 1998 ca și ISO 14882:1998, versiunea curentă fiind din 2003, ISO 14882:2003. Următoarea versiune standard, cunoscută informal ca C++0x, este în lucru. Limbajul de programare C++ a fost inițial derivat din C. Totuși, nu absolut orice program scris în C este valid C++. Deoarece C și C++ au evoluat independent, au apărut, din nefericire, o serie de incompatibilități între cele două limbaje de programare . Cea mai amplă revizie a limbajui C, C99, a creat un număr suplimentar de conflicte. Diferențele fac să fie greu de scris programe și biblioteci care să fie compilate și să ruleze corect în calitate de cod C sau C++, și produce confuzii celor care programează în ambele limbaje. Diferențele fac ca fiecare din cele două limbaje de programare să împrumute din caracteristicile celuilalt limbaj de programare.

Un **mediu de dezvoltare** (engl. software development environment, sau integrated development environment - "mediu integrat de dezvoltare) este un set de programe care ajută programatorul în scrierea programelor. Un mediu de dezvoltare combină toți pașii necesari creării unui program (ex.: editarea codului sursă, compilarea, depanarea, testarea, generarea de documentație) într-un singur soft, care, de regulă, oferă o interfață cu utilizatorul grafică, prietenoasă. Principalele componente ale unui mediu de dezvoltare sunt editorul de cod sursă și depanatorul. Mediile de dezvoltare apelează compilatoare, sau interpretoare, care pot veni în același pachet cu mediul însuși, sau pot fi instalate separat de către programator. Printre facilitățile prezente în mediile de dezvoltare mai sofisticate se numără: exploratoare de cod sursă, sisteme de control al versiunilor, designere de interfețe grafice, sau unelte de ingineria programării. De obicei un mediu de dezvoltare este specific unui anumit limbaj de programare, însă există la ora actuală și medii de dezvoltare care pot lucra cu mai multe limbaje, de ex. Eclipse sau Microsoft Visual Studio. În proiectul current, mediul în care am dezvoltat aplicaţia este **Code::Blocks,** care este un mediu de dezvoltare integrat pentru C/C++.

### 2.2 DIAGRAMA DE CLASE



**CAPITOLUL III**

#### 3. ANALIZA SOLUŢIEI IMPLEMENTATE

#### 3.1 FORMATUL DATELOR DE I/O

Prelucrarea datelor cu ajutorul fișierelor se face cu ajutorul clasei ifstream pentru citire, respectiv ofstream pentru scriere.Clasele ifstream si ofstream sunt derivate din clasa iostream .În momentul lansării în execuție a unui programce include iostream.h sunt create și inițializate automat 4 obiecte:

* cin –gestionează intrarea de la intrarea standard(tastatura);
* cout- gestionează ieșirea către ieșirea standard(monitrul);
* clog- gestionează ieșirea către dispozitivul standard de eroare(monitorul), utilizând buffer-e;
* cerr-gestionează ieșirea către dispozitivul standard de eroare(monitorul), neutilizând buffer-e.

Sintaxa constructorului clasei isftream este:ifstream Var(char \*NumeFisier, int mod=ios :: in, int buf = filebuf::openprot).Sintaxa constructorului clasei osftream este:ofstream Var(char \*NumeFisier, int mod= ios :: mod) Cel de al doilea parametru al constructorilor este opțional și specifică modul de deschidere al fișierului.

- ios::append - adaugă la sfârşitul fişierului;

- ios::atend - poziţionează pointer-ul la sfârşitul fişierului, însă informaţiile pot fi scrise oriunde in cadrul fişierului;

- ios::truncate - este modul de deschidere implicit: vechiul conţinut al fişierului este pierdut;

- ios::nocreate – dacă fişierul nu există, atunci operaţia eşuează;

- ios::noreplace - dacă fişierul deja există, atunci operaţia eşuează.

De obicei se folosesc variante prescurtate, și anume ate pentru attend, app pentru append, trunc pentru tuncate. Pentru a închide aceste fișiere trebuie apelată funcția close(). Rezultatul operaţiilor de intrare/ieşire poate fi testat prin intermediul funcţiilor

- eof() verifică dacă s-a ajuns la sfârşitul fişierului;

- bad() verifică dacă s-a executat o operaţie invalidă;

- fail() verifică dacă ultima operaţie a eşuat;

- good() verifică dacă toate cele trei rezultate precedente sunt false.

Poziţionarea în cadrul unui fişier se face prin intermediul funcţiei:

- seekg(int deplasare, int ref) unde ref reprezintă referinţa faţă de care se face deplasarea şi anume: ios::beg, ios::end, ios::cur, acesta fiind şi un argument având valoarea implicită ios::cur. Aflarea poziţiei curente în cadrul unui fişier se poate face prin intermediul funcţiei:

- tellg( ), tellp( ) ce va returna poziţia curentă pentru un flux de intrare şi respectiv pentru un flux de ieşire.

Exemplu de scriere într-un fișier text:

#include <iostream>

using namesapace std;

int main()

{

ofstram f(“mihai.txt”);

if(!f.fail())

{

f<<”M.I\n”;

f<<”Calculatoare,3121”;

f.close();

}

else

cout<<”Eroare!”;

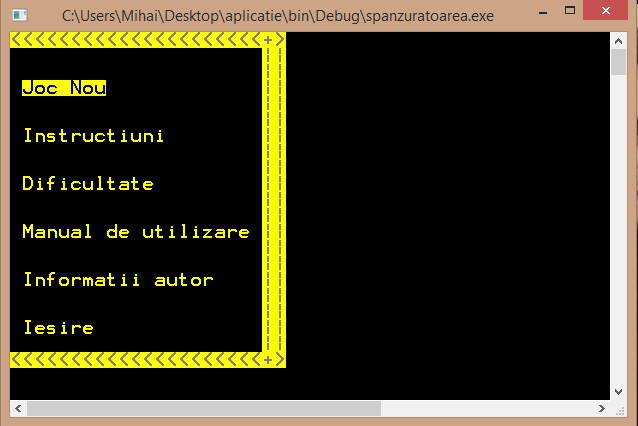
return 1;

}

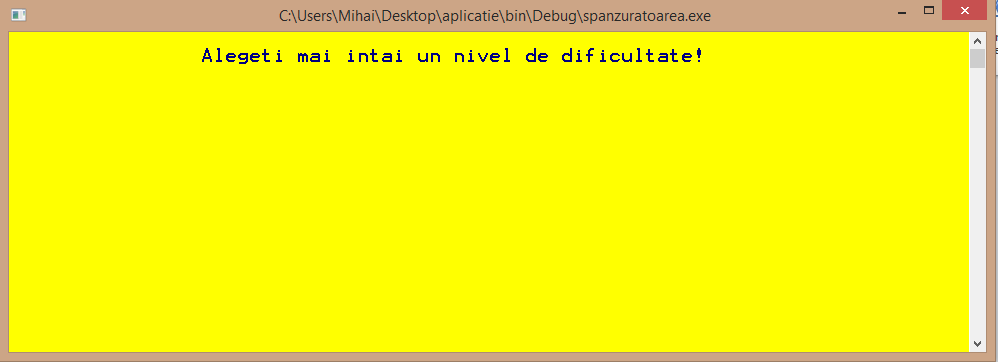
#### 

#### 3.2 STUDII DE CAZ

Meniu Joc Spanzuratoarea:



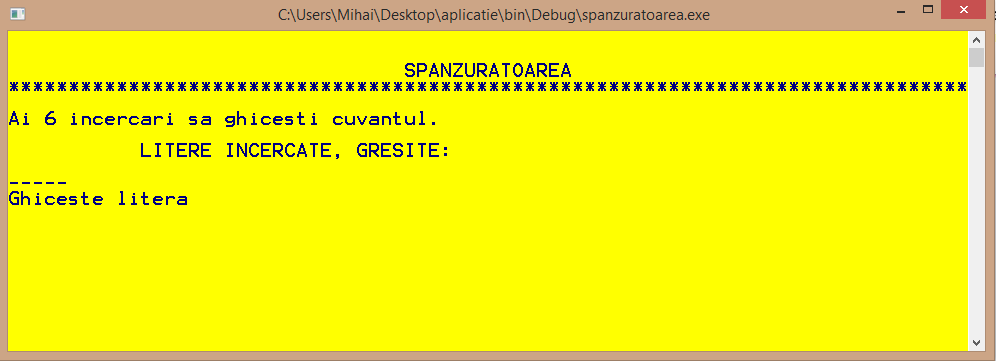
* Pentru a începe un joc nou, trebuie să selectăm Joc Nou ; iar dacă vom selecta Joc Nou, se va afişa urmatorul mesaj:



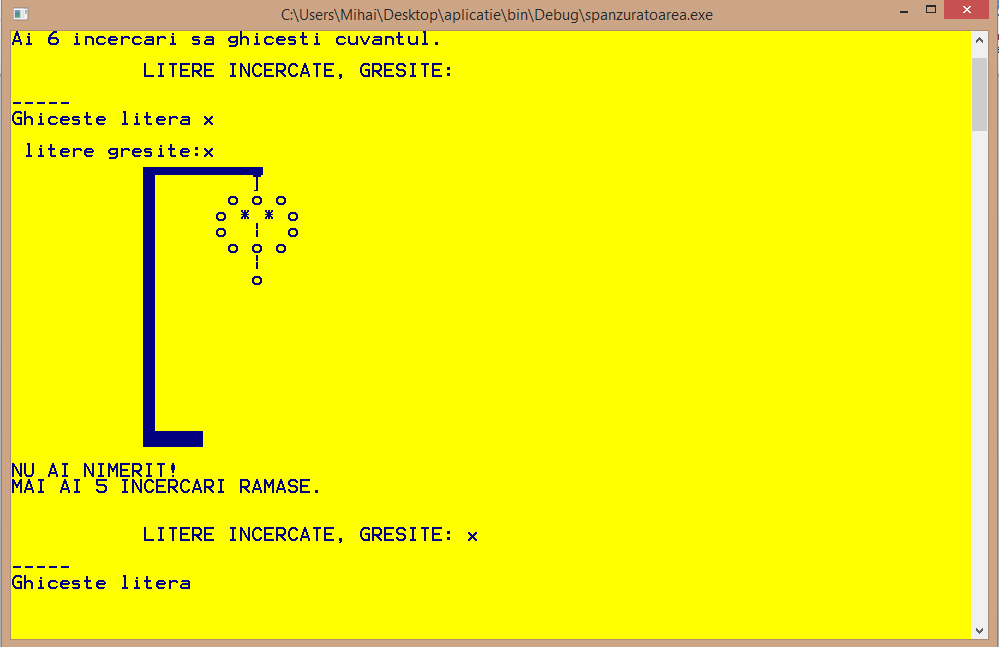
* Penru a alege un nivel de dificultate trebuie să selectăm Dificultate; mesajul va fi următorul:



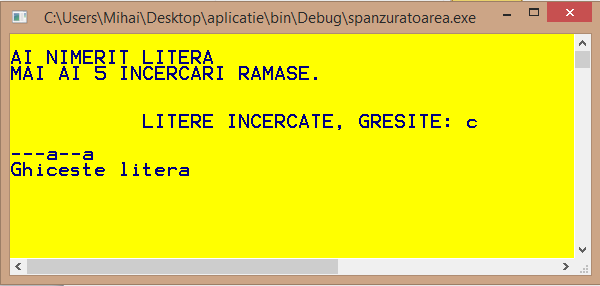
* Vom selecta spre exemplu primul nivel de dificultate, adică uşor. Vom începe un joc nou şi se va afişa:



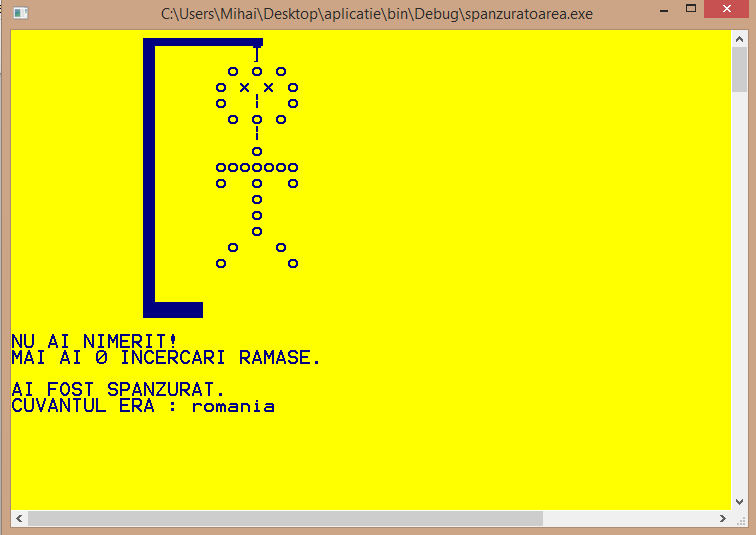
* Dacă utilizatorul va introduce o literă, care este greşită (spre exemplu ”x”), se va afişa:



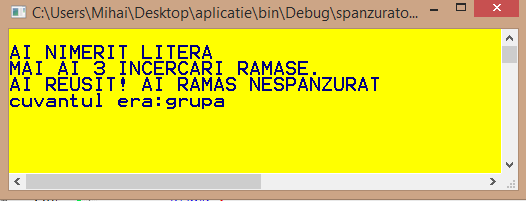
* Dacă utilizatorul va intoduce o literă care este corectă (spre exemplu “a”), se va afişa:



* Dacă utilizatorul va epuiza toate încercarile puse la dispoziţie şi cuvântul nu a fost ghicit, se va afişa urmatorul mesaj:



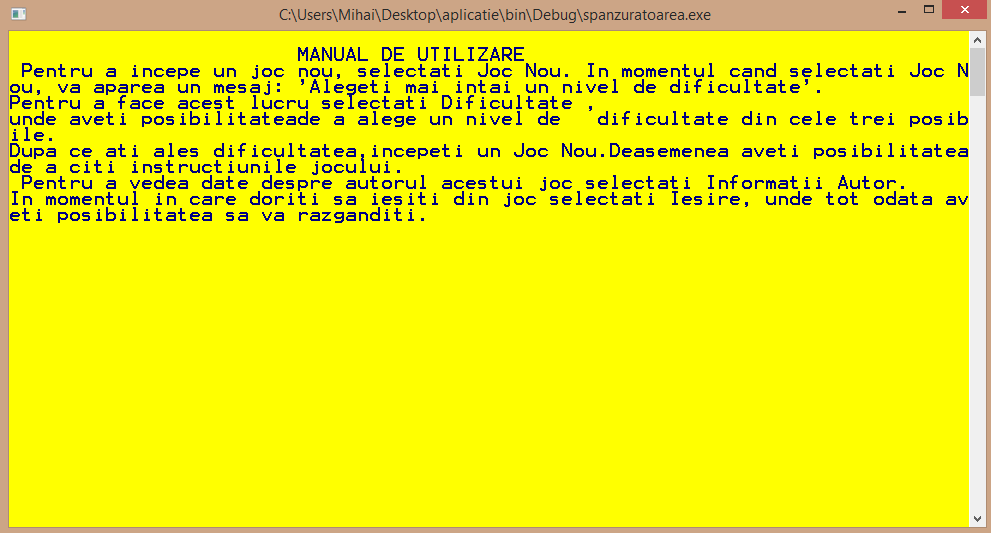
* Dacă utilizatorul va ghici cuvâtul pâna la epuizarea numarului de încercări atunci va primi urmatorul mesaj:



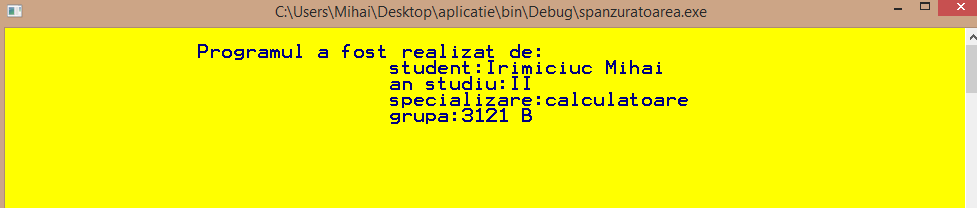
* Dacă utilizatorul doreşte să citească instrucţiunile jocului, trebuie sa selecteze Instrucțiuni şi i se va afişa:



* Pentru ca utilizatorul să citească manualul de utilizare al jocului va trebui să selecteze Manual de Utilizare şi se va afişa:



* Pentru a vedea date despre autorul programului se va selecta Informații Autor



* Dacă utilizatorul doreşte sa parăsească jocul, trebuie să selecteze Iesire .

**CAPITOLUL IV**

##### 4.CONCLUZII

Proiectul prezentat este construit în mediul de utilizare Code::Blocks, folosind limbajul de programare C++ prin tehnologiile specific Programării Orientate pe Obiecte. Dintre tehnologiile POO s-au folosit: clase, moştenirea, transmiterea parametrilor prin valoare.

Proiectul conţine o aplicaţie joc şi anume joculeţul spânzurătoarea. Acest joc presupune ca jucătorul să ghicească un cuvânt, cuvânt ce va fi ales random dintr-una din listele string de cuvinte din aplicaţie. Aplicaţia dispune de trei liste de cuvinte, corespunzătoare celor trei nivele de dificultate din joc: uşor, mediu şi greu. Meniul aplicaţiei dispune de şase opţiuni şi anume: JOC NOU, DIFICULTATE, INSTRUCŢIUNI, MANUAL DE UTILIZARE, DATE DESPRE AUTOR şi IEŞIRE.

Aşadar, utilizatorul işi poate alege dificultatea jocului, care de altfel este o opţiune obligatorie pentru a începe un joc nou, poate citi instrucţiunile, manualul de utilizare şi datele despre autorul programului şi poate părăsi jocul prin opţiunea ieşire.

Aplicaţia prezentată este o aplicaţie care pare destul de simplă, dar care a necesitat atenţie şi muncă, iar ca posibile îmbunataţiri aduse aplicaţiei, pot menţiona construirea unei funcţii care să ofere un timp limitat (un cronometru) utilizatorului pentru fiecare mişcare, si imbunatatiri aduse la grafică.

**CAPITOLUL V**

###### 5.BIBLIOGRAFIE

1. [http://www.cplusplus.com](http://www.cplusplus.com/)
2. [Wikipedia](http://www.wikipedia.org)
3. <http://apollo.eed.usv.ro/~remus/>
4. Suport curs „Programare Orientată pe Obiecte”, anul II, Prof. Dr. Ing. Prodan Remus;
5. Suport laborator „Programare Orientată pe Obiecte”, anul II, Prof. Dr. Ing. Prodan Remus;