Colegiul Național de Informatică

Piatra-Neamț

**Proiect pentru atestarea competențelor profesionale la informatică**

**Biblioteca**

**Basic Function Library**

|  |  |
| --- | --- |
| Profesor Coordonator: Gabriela Blaga | Elev: Savin Mihnea |
|  |  |
|  | Clasa: a-XII-a E |

Mai 2024

Cuprins

MOTIVUL ALEGERII TEMEI DE PROIECT 2

METODE DE REALIZARE 3

1. IDE 3

1.Crearea unui proiect nou in Visual Studio 4

2.Interfața Visual Studio 6

2. C++ 9

1.Sintaxa 9

*1.Sintaxa de bază a unui program C++* 10

*2.Fișierele Header* 10

*3.Namespace* 11

*3.Funcția main()* 11

*4.Blocuri* 12

*5.Punctul și virgula* 12

*6.Identificatori* 12

*7.Cuvinte cheie* 13

*8. Scrierea folosind ”cout”* 13

2.Programarea orientată pe obiecte 14

*1.Clase* 14

*2.Membrii unei clase – Membrii de date și funcții membre* 15

*3.Obiecte* 15

MOD DE IMPLEMENTARE 16

1.Crearea proiectului 16

1.Biblioteci în c++ 16

2.Linking 17

3.Biblioteci Statice și Dinamice 19

1.Legătura statică 19

2.Legătura dinamică 20

4.Alte setări ale proiectului 21

2.Scrierea codului 22

1.Fișierul header 22

2.Fișierul sursă 23

3.Exportarea bibliotecii 24

Codul sursă 25

1.Fișierul header 25

2.Fișierul sursă 26

Concluzii 32

BIBLIOGRAFIE 33

# MOTIVUL ALEGERII TEMEI DE PROIECT

Tema proiectului este crearea unei biblioteci de funcții și metode des întâlnite și folosite mai ales de elevii de liceu, dar nu numai. Am denumit acest proiect ’’Basic Function Library’’ sau, pe scurt, BSC, un nume simplu ce sugerează scopul acestui proiect.

Am ales această temă din mai multe motive:

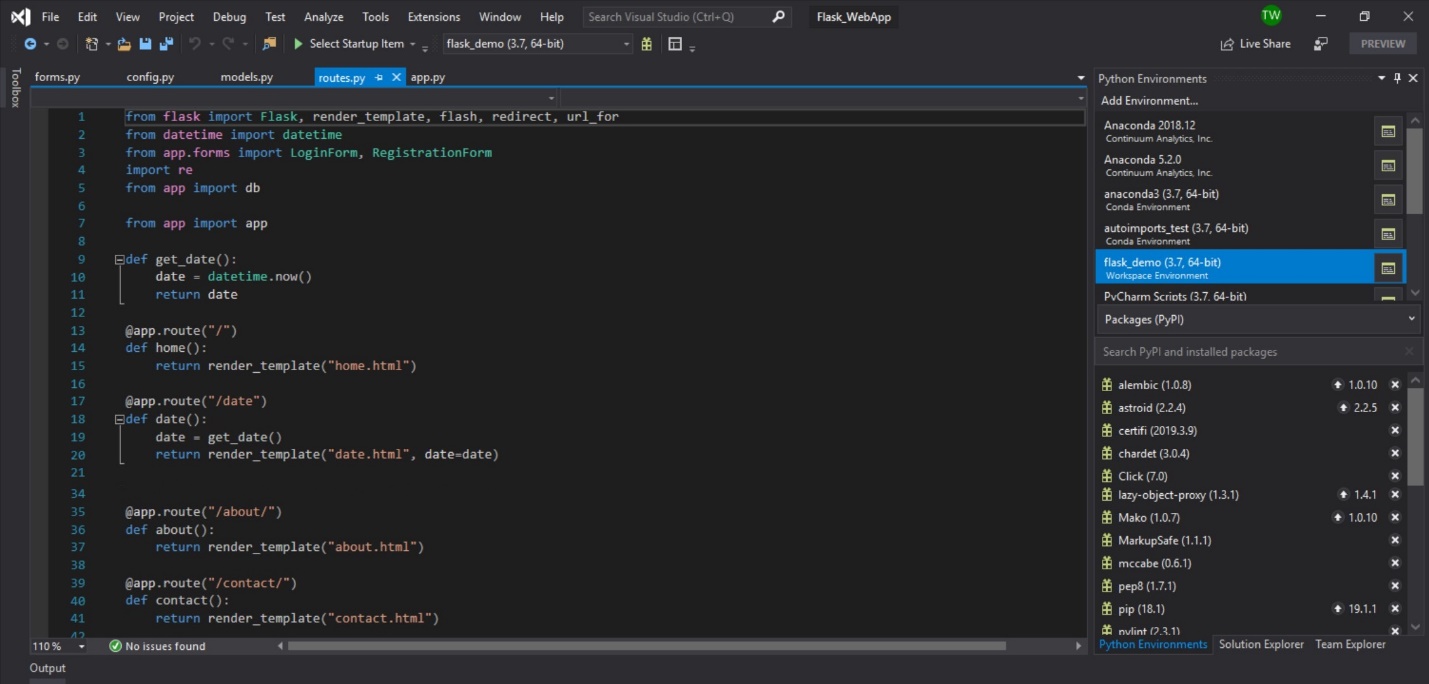
Un prim motiv este dorința de a câștiga timp și de a scăpa de situațiile în care trebuie să rescriu aceeași funcție de mai multe ori în diferite locuri oricând rezolv o problemă sau lucrez la un proiect și am nevoie de o metodă ce efectuează lucruri de bază, cum ar fi o citire, o sortare sau alte operații des întâlnite în informatică. Rescrierea durează timp pe care, cu ajutorul unei biblioteci, l-aș câștiga și investi către rezolvarea efectivă problemei pe care o am.

Cel de-al doilea motiv pentru care am ales să realizez acest proiect este curiozitatea mea față de modul de funcționare a unei biblioteci și posibilitatea de a o instala și folosi la orice proiect, pe orice calculator, cu ușurință. Astfel, prin realizarea acestui proiect m-am întâlnit cu diferite concepte cum ar fi tipurile de alocare a unei biblioteci, linker-ul C++ și instalarea unei biblioteci externe într-un proiect, definirea unui namespace, lucrul cu clase, generalizarea metodelor pentru a putea fi folosite în cât mai multe cazuri particulare.

# METODE DE REALIZARE

1. IDE 

Un mediu de dezvoltare sau IDE (integrated development environment - „mediu integrat de dezvoltare”) este un set de [programe](https://ro.wikipedia.org/wiki/Software) care ajută programatorul în scrierea programelor. Un mediu de dezvoltare combină toți pașii necesari creării unui program (ex.: editarea [codului sursă](https://ro.wikipedia.org/wiki/Cod_surs%C4%83), [compilarea](https://ro.wikipedia.org/w/index.php?title=Compilare&action=edit&redlink=1), [depanarea](https://ro.wikipedia.org/w/index.php?title=Depanare_(informatic%C4%83)&action=edit&redlink=1), [testarea](https://ro.wikipedia.org/w/index.php?title=Testare&action=edit&redlink=1), generarea de [documentație](https://ro.wikipedia.org/wiki/Documenta%C8%9Bie)) într-un singur soft, care, de regulă, oferă o interfață cu utilizatorul grafică, prietenoasă. Câteva exemple de medii de dezvoltare sunt: Code::Blocks, Microsoft Visual Studio, NetBeans, Eclipse, etc..

Pentru acest proiect am folosit Microsoft Visual Studio deoarece este un mediu integrat de dezvoltare modern, ușor de folosit și inovativ. 

(Imagine exemplu cu interfața Visual Studio)

### 1.Crearea unui proiect nou in Visual Studio

A screenshot of a computer

Description automatically generated

La deschiderea programului suntem întâmpinați de fereastra de mai sus. Pentru a crea un proiect nou alegem ”Create a new project”. În următoarea fereastră trebuie să alegem tipul de proiect pe care vrem să îl creăm, în funcție de nevoile fiecăruia. Aceste tipuri de proiecte (templates) trebuie instalate din managerul de instalare.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Pentru a instala sau modifica pachetele existente selectăm butonul ”Modify” corespunzător versiunii pe care dorim să o folosim. În imaginea atașată există o singură versiune instalată.

A screenshot of a computer

Description automatically generated

În fereastra aferentă butonului ”Modify” putem instala diferite pachete pentru mai multe limbaje de programare sau pentru dezvoltarea aplicațiilor pentru alte platforme. În partea dreaptă a ferestrei putem alege exact ce componente ale pachetelor dorim să instalăm și mai jos spațiul pe care îl ocupă în memorie.

A screenshot of a computer

Description automatically generated După selectarea tipului de proiect trebuie să îl denumim, să alegem locația unde va fi salvat și apoi putem apăsa butonul ”Create”.

### 2.Interfața Visual Studio

A screenshot of a computer

Description automatically generated

Aceasta este interfața care ne întâmpină după ce apăsăm butonul ”Create”.

Pentru a putea scrie cod trebuie să creăm un fișier .cpp. Putem face acest lucru prin File → New → File.

A screenshot of a computer

Description automatically generated

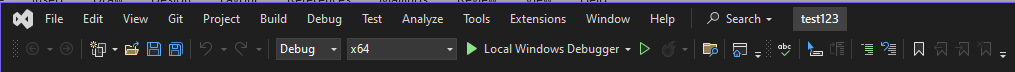
A screenshot of a computer

Description automatically generated Mai departe suntem întâmpinați de fereastra din dreapta. În partea stângă alegem tipul de fișier de care avem nevoie, în acest caz Visual C++. Apoi selectăm C++ file și putem începe scrierea codului.

+

Putem găsi, de asemenea, multe alte funcționalități în meniul din partea de sus. De exemplu, butonul verde plin, în formă de săgeată, este butonul de compilare care depanează în timpul compilării (Local Windows Debugger), iar cel de lângă, care are doar conturul verde, compilează codul fără al depana (run without debugging).

Un alt buton important este ”Active solution platform”, se găsește în stânga butonului ”Local Windows Debugger” și oferă opțiunea de a alege pe câți biți va funcționa compilatorul.



A screenshot of a computer

Description automatically generated

În partea stângă se mai găsesc și butoanele pentru accesarea altor foldere, crearea unui nou proiect și salvare.

În partea de sus se mai multe meniuri (File, Edit, View, etc.) de unde putem schimba diferite setări, crea file, lucra pe un proiect Github și alte funcționalități.

A screenshot of a computer program

Description automatically generatedPentru a accesa setările proiectului, apăsăm click dreapta pe numele acestuia în ”Solution Explorer” și apoi Properties.

Această fereastră aduce diferite opțiuni si setări ale proiectului ce pot modificate. De exemplu aici putem seta linker-ul astfel încât să legăm proiectul de o bibliotecă externă.

1. C++



C++ este un [limbaj de programare](https://ro.wikipedia.org/wiki/Limbaj_de_programare) general, [compilat](https://ro.wikipedia.org/wiki/Compilator). Este un [limbaj multi-paradigmă](https://ro.wikipedia.org/wiki/Limbaj_de_programare_multi-paradigm%C4%83), cu verificarea statică a tipului variabilelor ce suportă [programare procedurală](https://ro.wikipedia.org/wiki/Programare_procedural%C4%83), [abstractizare a datelor](https://ro.wikipedia.org/w/index.php?title=Abstractizare_(informatic%C4%83)&action=edit&redlink=1), [programare orientată pe obiecte](https://ro.wikipedia.org/wiki/Programare_orientat%C4%83_pe_obiecte). În [anii 1990](https://ro.wikipedia.org/wiki/Anii_1990), C++ a devenit unul dintre cele mai populare limbaje de programare comerciale, rămânând astfel până azi.

A screen shot of a computer code

Description automatically generated

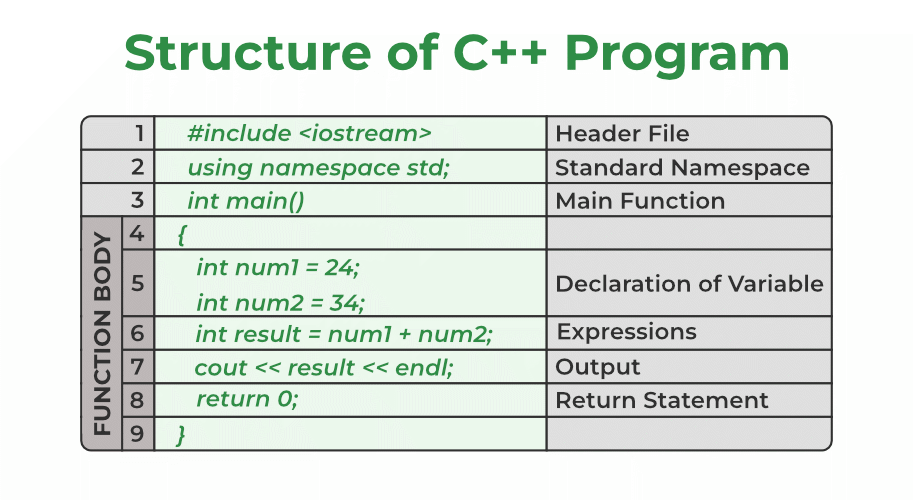
(Exemplu de program ce afișează “Hello World!)

### 1.Sintaxa

Sintaxa se referă la regulile și reglementările de scriere a codului în cadrul unui limbaj de programare. Ea poate fi văzută și ca setul de reguli gramaticale ce definește structura unui limbaj de programare.

Limbajul C++ are, deci, propriile reguli de sintaxă.

#### *1.Sintaxa de bază a unui program C++*

Sintaxa de bază a unui program:

(figura 1)

#### *2.Fișierele Header*

Fișierele header conțin definiția funcțiilor pe și macro-urilor pe care urmează să le folosim. Instrucțiunea ”#include <iostream>” transmite compilatorului să includă fișierul header al bibliotecii iostream care conține definiția metodelor ”cin” și ”cout”, folosite pentru citirea datelor de la tastatură, respectiv scrierea datelor pe ecran. Această instrucțiune se mai găsește și în forma ” #include ”numeBibliotecă.lib” ” sau ” #include ”numeBibliotecă.dll” ”, ghilimelele sugerând faptul că biblioteca este externă, nu este preinstalată în compilator.

#### *3.Namespace*

Namespace-urile sunt folosite pentru a crea un domeniu unde se pot defini identificatori. Se folosesc pentru a evita confuziile între doi identificatori știind că identificatorii trebuie să aibă denumiri unice.

În a doua linie din imagine, instrucțiunea ”using namespace std;” specifică faptul că toți identificatorii care au același nume cu cei definiți în namespace-ul standard vor fi considerați ca membrii ai acestuia. De exemplu, nu mai este nevoie să scriem ”std ::” înainte de metode ce fac parte din namespace-ul standard cum ar fi ”min” sau ”max” (std :: min). Însă această instrucțiune vine și cu dezavantajele ei. Folosind exemplul anterior, nu am mai putea declara variabile cu numele ”min” sau ”max” deoarece s-ar suprapune cu identificatorii definiți în std. Pentru ca acest lucru să fie posibil trebuie să omitem această instrucțiune și să folosim ”std ::” înaintea oricărui identificator din standard namespace de fiecare dată când folosim unul.

#### *3.Funcția main()*

Funcțiile sunt cărămizile de bază ale unui program în C++ și conțin instrucțiunile pentru îndeplinirea unei sarcini specifice. Înafară de instrucțiunile pe care le conține, o funcție are și un antet (definiție), care conține informații despre tipul ei (dacă și ce tip de date returnează) și parametrii acesteia.

A screen shot of a computer

Description automatically generatedÎn a treia linie este definită funcția ”int main()”. Funcția main este cea mai importantă funcție în orice program C++. Execuția oricărui program începe din funcția main, toate celelalte funcții fiind apelate înăuntrul acesteia, iar aceasta trebuie să returneze o valoare ce indică starea executării.

#### *4.Blocuri*

Blocurile sunt un grup de instrucțiuni închise între acolade (”{” , ”}”). Acestea definesc domeniile identificatorilor și sunt folosite, în general, pentru a defini corpurile funcțiilor sau structurilor de control.

Între liniile 4 și 9 (figura 1) se află corpul funcției main, închis între acolade.

#### *5.Punctul și virgula*

Punctul și virgula se folosește pentru a indica sfârșitul unei instrucțiuni. Când compilatorul găsește punctul și virgula încheie operația și trece la următoarea linie.

#### *6.Identificatori*

Identificatorii se folosesc pentru a denumi variabile, funcții, obiecte, și alte tipuri de date definite de dezvoltatori (ex. struct). În linia 5 (fig. 1) sunt declarate două variabile: int este tipul de dată (integer – număr întreg), iar num1 și num2 sunt identificatorii acestor variabile.

Un identificator poate fi format din litere mici și mari, underscore (”\_”) și numere cu condiția ca primul caracter să nu fie număr.

#### A screenshot of a computer screen Description automatically generated*7.Cuvinte cheie*

În C++ există un set de cuvinte ce au un sens special in cadrul unui program. Acestea nu pot fi folosite pe post de identificatori.

De exemplu, cuvintele ”int”, ”return” și ”using” sunt cuvinte cheie folosite in programul exemplu (figura 1). În C++ există 95 de cuvinte cheie.

#### *8. Scrierea folosind ”cout”*

În linia 7 este folosită metoda ”cout” pentru a scrie pe ecran suma a două numere (num1 , num2) – ”cout << result << endl;”. Instrucțiunea ”endl” trece scrierea la un rând nou.

### 2.Programarea orientată pe obiecte

Limbajul de programare C++ suportă, pe lângă programarea procedurală, programarea orientată spre obiecte. Exemplul de mai devreme (figura 1) reprezintă programarea procedurală.

A close-up of a table

Description automatically generated Aici este un exemplu de programare orientată pe obiecte:

(figura 2)

#### *1.Clase*

O clasă este șablonul unui obiect (ex. Fie ”animal” o clasă. Câinele este un obiect al acestei clase). O clasă are proprii membrii de date și propriul comportament (funcțiile membre). Prima literă din denumirea unei clase trebuie să fie mare și este folosit cuvântul cheie pentru declararea acesteia (figura 2, linia 3).

#### *2.Membrii unei clase – Membrii de date și funcții membre*

Atributele sau datele dintr-o clasă sunt definite de membrii de date, iar funcțiile membre lucrează cu aceste date.

În linia 5 (figura 2) sunt declarați membrii de date ”num1” și num2”, iar in linia 6 este definită funcția membră ce prelucrează aceste date.

Membrii unei clase pot fi de 2 tipuri:

* Membrii privați: pot fi accesați și prelucrați doar înăuntrul clasei de alți membri ai acesteia;
* Membri publici: pot fi accesați și prelucrați de oriunde în program.

#### *3.Obiecte*

Un obiect este o instanță a unei clase. Clasa este doar un șablon căruia nu îi este alocată memorie. Pentru a folosi membrii unei clase trebuie să creăm un obiect al acelei clase.

Observație: Dacă o clasă are funcții membre declarate publice nu este nevoie de un obiect al acelei clase pentru a fi apelată dacă funcția nu are nevoie de unul pentru a funcționa.

Un obiect se declară asemenea unei variabile. (ex. ”NumeClasă numeObiect”). Pentru a accesa datele și metodele acestui obiect se folosește operatorul ”.” (ex. ”numeObiect.numeFunctie()”).

# MOD DE IMPLEMENTARE

## 1.Crearea proiectului

Pașii pe care i-am urmat pentru a crea un proiect sunt identici cu cei prezentați mai devreme când am prezentat mediul de dezvoltare. Șablonul proiectului este ”Static Library”:

A screenshot of a computer

Description automatically generated

### 1.Biblioteci în c++

În informatică, o bibliotecă este o colecție de resurse nevolatile în memorie folosite adesea de programe sau în dezvoltarea programelor. Acestea pot include date de configurare, documentații, cod ”prescris” (pentru a evita situațiile în care acesta este refolosit, se repetă de multe ori), subrutine, clase, etc.

Scopul principal al bibliotecii prezentate în acest proiect e de a cuprinde cât mai mult cod ”prescris”, de a acoperi cât mai multe subrutine des folosite, de obicei, de elevii de liceu, dar nu numai.

Codul unei biblioteci e organizat astfel încât să poată fi folosit în programe multiple care nu au nicio legătură între ele, în timp ce codul acelor programe este organizat astfel încât să poată fi folosit doar în cadrul acestuia. Astfel, valoarea unei biblioteci constă în refolosirea unor elemente standardizate. Atunci când un program invocă o bibliotecă acesta primește toate funcționalitățile acelei biblioteci fără obligația de a le implementa.

### 2.Linking

Procesul de linking (legare) între bibliotecă și program este esențial în funcționarea acestuia, proces realizat automat de linker sau de compilator. Însă, ca acest proces să poată avea loc, trebuie să specificăm linker-ului unde se află în memorie biblioteca pe care vrem să o includem. În cazul meu, am realizat acest lucru prin pașii următori:

* Am accesat setările proiectului programului test (pag. 10)
* A screenshot of a computer program

  Description automatically generatedLa setarea C/C++→General→Additional Include Directories am adăugat locația în memorie a fișierului header;
* A screenshot of a computer program

  Description automatically generatedLa setarea Linker→General→Additional Library Directories am adăugat locația în memorie a bibliotecii;
* La setarea Linker→Input→Additional Dependencies am adăugat numele bibliotecii.

A screenshot of a computer

Description automatically generated

### 3.Biblioteci Statice și Dinamice

Există două tipuri de legături între biblioteci și program: statice și dinamice.

Legătura statică presupune legarea programului de bibliotecă la compilare, rezultând un singur fișier executabil ce poate fi executat cu ușurință.

Legătura dinamică nu implică nicio legătură de cod. În schimb necesită ca biblioteca să fie încărcată în memorie înainte ca programul să poată rula. Acest lucru duce la o mai bună gestionare a memoriei deoarece doar componentele necesare sunt încărcate atunci când este nevoie ceea ce duce la niveluri mai ridicate de eficiență generală.

Fiecare din aceste două tipuri au avantajele și dezavantajele lor:

#### 1.Legătura statică

Avantaje:

* Timp de execuție mai scurt
* Gestionare bună a memoriei (toate modulele sunt legate direct între ele)
* Ușurința la implementare
* Control mai bun asupra arhitecturii programului
* Execuția ieftină (cantitatea redusă de resurse necesare la rulare)
* Evitarea duplicării codului

Dezavantaje:

* Depanare mai dificilă
* Compatibilitate limitată între biblioteci
* Securitate redusă în comparație cu legătura dinamică
* Nu poate fi modificată în timpul execuției
* Incompatibilitate cu anumite biblioteci
* Dificil de întreținut

#### 2.Legătura dinamică

Avantaje:

* Timp de încărcare mai rapid
* Scalabilitate îmbunătățită
* Întreținere ușoară
* Separarea codului
* Portabilitate mai bună
* Cerințe reduse de memorie

Dezavantaje:

* Fragilitate
* Probleme de securitate (risc sporit la injecția de cod rău intenționat când vine vorba de biblioteci externe)
* Probleme de dependențe
* Performanță redusă
* Versiuni nesigure
* Formate incompatibile la migrarea între sisteme

Pentru acest proiect am ales să folosesc legătura statică datorită ușurinței la implementare, evitarea duplicării codului și execuția rapidă și eficientă.

### 4.Alte setări ale proiectului

A screenshot of a computer

Description automatically generated Prima setare esențială la implementare este compilarea programului pe 32 de biți.

În setările proiectului la secțiunea ”Platform” am ales ”All Platforms”.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

Dacă sunt deschise mai multe proiecte în ”Solution Explorer” trebuie să alegem proiectul care conține funcția ”main()” și poate fi deci compilat. (Solution ”ProjectName” (x of x projects) →Properties). La setarea ”Single startup project” am ales proiectul unde se pot testa funcționalitățile bibliotecii.

A screenshot of a computer

Description automatically generated

## 2.Scrierea codului

### 1.Fișierul header

### 

A screenshot of a computer

Description automatically generated

Pentru a începe scrierea codului este nevoie de două fișiere. Mai exact un fișier header și un fișier sursă.

A screenshot of a computer program

Description automatically generated

În cadrul fișierului header am declarat un namespace ”BSC”. Înăuntrul acestui namespace am declarat mai multe clase pentru a ordona metodele în funcție operația pe care o efectuează.

A computer screen shot of a program code

Description automatically generated(ex. Clasa ”Array” și unele dintre metodele pe care le conține)

### 2.Fișierul sursă

A screenshot of a computer

Description automatically generated Fișierul sursă începe cu operațiile de ”include”. Se includ fișierul header al bibliotecii și alte biblioteci sau module de care este nevoie.

 Apoi am definit namespace-ul ”BSC”.

A computer screen shot of a program code

Description automatically generated Înăuntrul acestui namespace se află metodele declarate în fișierul header și codul lor sursă (ex. de metodă din fișierul sursă).

## 3.Exportarea bibliotecii

A screenshot of a computer program

Description automatically generated Pentru a exporta biblioteca și a o folosi trebuie să o construim. Pentru a face asta trebuie să mergem la meniul Build→Build Solution sau apăsăm butonul F7.

 Prin acest proces se va construi fișierul .lib care poate fi exportat. (importarea unei biblioteci pag. 19 – linking)

# Codul sursă

## 1.Fișierul header

#pragma once

namespace BSC

{

class Reader

{

public:

static void ArrayReader(int[], int);

static int\*\* PD2ArrayReader(int, int);

static int\*\* PD2SArrayReader(int);

};

class Writer

{

public:

static void ArrayWriter(int[], int);

static void ArrayWriter\_ns(int[], int);

};

class Mathematics {

public:

static long long int GaussSum(int);

static int DigitSum(int);

static int Factorial(int);

static void DigitTable(int[], int);

static int NumLength(int);

static bool IsPrime(int);

static int GreatestCommonDivisor(int, int);

static void SieveOfEratosthenes(int[], int);

static int MaxDigit(int);

static int MinDigit(int);

static int ControlDigit(int);

static void Fibonacci(int[], int);

static int RusseMultiplication(int, int);

static int FastExponentiation(int, int);

static double GoldenRatio();

};

class Array {

public:

static void DeleteItem(int[], int&, int);

static void AddItem(int[], int&, int, int);

static void BubbleSort(int[], int);

static void SelectionSort(int[], int);

static void InsertionSort(int[], int);

static void QuickSort(int[], int, int);

static int BinarySearch(int[], int, int, int);

static int MaxNum(int[], int);

static int MinNum(int[], int);

};

} // end namespace

## 2.Fișierul sursă

#include <iostream>

#include <cmath>

#include "BSC.h"

namespace BSC

{

void Reader::ArrayReader(int v[], int n)

{

for (int i = 0; i < n; i++)

std::cin >> v[i];

}

int\*\* Reader::PD2ArrayReader(int n, int m) //Only works for int\*\*

{

int\*\* a = 0;

a = new int\* [n];

for (int i = 0; i < n; i++)

{

a[i] = new int[m];

for (int j = 0; j < m; j++)

std::cin >> a[i][j];

}

return a;

}

int\*\* Reader::PD2SArrayReader(int n)

{

int\*\* a = 0;

a = new int\* [n];

for (int i = 0; i < n; i++)

{

a[i] = new int[n];

for (int j = 0; j < n; j++)

std::cin >> a[i][j];

}

return a;

}

void Writer::ArrayWriter(int v[], int n)

{

for (int i = 0; i < n; i++)

std::cout << v[i] << ' ';

}

void Writer::ArrayWriter\_ns(int v[], int n)

{

for (int i = 0; i < n; i++)

{

if (i == n - 1)

std::cout << v[i];

else std::cout << v[i] << ' ';

}

}

long long int Mathematics::GaussSum(int n)

{

if (n % 2) return 1LL \* Mathematics::RusseMultiplication(n, (n + 1) / 2);

else return 1LL \* Mathematics::RusseMultiplication(n / 2, n + 1);

}

int Mathematics::DigitSum(int number)

{

int sum = 0;

while (number)

{

sum += number % 10;

number /= 10;

}

return sum;

}

int Mathematics::Factorial(int n)

{

if (n == 1)

return n;

return n \* Factorial(n - 1);

}

void Mathematics::DigitTable(int v[], int number)

{

int iterator = NumLength(number) - 1;

while (iterator > -1)

{

v[iterator--] = number % 10;

number /= 10;

}

}

int Mathematics::NumLength(int number)

{

int lenght = 0;

while (number)

{

number /= 10;

lenght++;

}

return lenght;

}

bool Mathematics::IsPrime(int number)

{

int divider;

if (number < 2)

return false;

if (number == 2)

return true;

if (number % 2 == 0)

return false;

for (divider = 3; divider \* divider < number; divider += 2)

if (number % divider == 0)

return false;

if (divider \* divider == number)

return false;

return true;

}

int Mathematics::GreatestCommonDivisor(int number1, int number2)

{

while (number1 != number2)

{

if (number1 > number2)

number1 -= number2;

else number2 -= number1;

}

return number1;

}

void Mathematics::SieveOfEratosthenes(int v[], int target)

{

for (int i = 0; i <= target; i++)

v[i] = 0;

v[0] = 1;

v[1] = 1;

for (int i = 2; i \* i <= target; i++)

{

if (v[i] == 0)

for (int j = 2; j <= target / i; j++)

v[i \* j] = 1;

}

}

int Mathematics::MaxDigit(int number)

{

int max = -1;

while (number)

{

if (number % 10 > max)

max = number % 10;

number /= 10;

}

return max;

}

int Mathematics::MinDigit(int number)

{

int min = 10;

while (number)

{

if (number % 10 < min)

min = number % 10;

number /= 10;

}

return min;

}

int Mathematics::ControlDigit(int number)

{

int controlDigit = 0;

while (number)

{

controlDigit += number % 10;

number /= 10;

}

if (controlDigit < 10)

return controlDigit;

else if (controlDigit % 10 == 0)

return controlDigit / 10;

else

return ControlDigit(controlDigit);

}

void Mathematics::Fibonacci(int v[], int target)

{

v[0] = 1;

v[1] = 1;

for (int i = 2; i < target; i++)

v[i] = v[i - 1] + v[i - 2];

}

int Mathematics::RusseMultiplication(int number1, int number2)

{

int result = 0;

while (number1)

{

if (number1 % 2)

result += number2;

number1 /= 2;

number2 \*= 2;

}

return result;

}

int Mathematics::FastExponentiation(int base, int exponent)

{

int p = 1;

while (exponent)

{

if (exponent % 2)

p \*= base;

exponent /= 2;

base \*= base;

}

return p;

}

double Mathematics::GoldenRatio()

{

return (double)((1 + sqrt(5)) / 2);

}

void Array::DeleteItem(int v[], int& n, int x)

{

int j;

for (int i = 0; i < n; i++)

if (v[i] == x)

{

j = i;

while (j < n - 1)

{

v[j] = v[j + 1];

j++;

}

n--;

i--;

}

}

void Array::AddItem(int v[], int& n, int poz, int item)

{

for (int i = n - 1; i >= poz; i--)

{

v[i + 1] = v[i];

}

v[poz] = item;

n++;

}

void Array::BubbleSort(int v[], int n)

{

bool sortat;

do {

sortat = true;

for (int i = 0; i < n - 1; i++)

{

if (v[i] > v[i + 1])

{

std::swap(v[i], v[i + 1]);

sortat = false;

}

}

} while (!sortat);

}

void Array::SelectionSort(int v[], int n)

{

for (int i = 0; i < n - 1; i++)

for (int j = i + 1; j < n; j++)

if (v[i] > v[j])

std::swap(v[i], v[j]);

}

void Array::InsertionSort(int v[], int n)

{

for (int i = 1; i < n; i++)

{

int p = i;

while (p > 0 && v[p] < v[p - 1])

{

std::swap(v[p], v[p - 1]);

p--;

}

}

}

void Array::QuickSort(int v[], int st, int dr) // dr == n - 1;

{

if (st < dr)

{

int m = (st + dr) / 2;

std::swap(v[st], v[m]);

int i = st, j = dr, d = 0;

while (i < j)

{

if (v[i] > v[j])

{

std::swap(v[i], v[j]);

d = 1 - d;

}

i += d;

j -= 1 - d;

}

QuickSort(v, st, i - 1);

QuickSort(v, i + 1, dr);

}

}

int Array::BinarySearch(int v[], int st, int dr, int target)

{

if (st > dr)

return -1;

int m = (st + dr) / 2;

if (v[m] == target)

return m;

else if (target < v[m])

return BinarySearch(v, st, m - 1, target);

else

return BinarySearch(v, m + 1, dr, target);

}

int Array::MaxNum(int v[], int n)

{

int max = INT\_MIN;

for (int i = 0; i < n; i++)

if (max < v[i])

max = v[i];

return max;

}

int Array::MinNum(int v[], int n)

{

int min = INT\_MAX;

for (int i = 0; i < n; i++)

if (min > v[i])

min = v[i];

return min;

}

}

# Concluzii

Tema acestui proiect mi-a deschis o nouă perspectivă asupra informaticii. Am învățat cum să îmi fac propria bibliotecă și să o folosesc în cadrul altor programe. De asemenea, am învățat despre tipurile de biblioteci si cum funcționează ele în cadrul unui program.

În concluzie, realizarea acestui proiect a fost o experiență valoroasă datorită căreia am descoperit un domeniu al informaticii care mă intrigă.

# BIBLIOGRAFIE

<https://ro.wikipedia.org/wiki/Mediu_de_dezvoltare>

<https://visualstudio.microsoft.com/>

<https://ro.wikipedia.org/wiki/C%2B%2B>

<https://www.geeksforgeeks.org/cpp-basic-syntax/>

<https://www.pbinfo.ro/>

<https://en.cppreference.com/w/cpp/keyword>

<https://en.wikipedia.org/wiki/Library_(computing)>

[https://www.ranktracker.com/ro/blog/static-vs-dynamic-linking- what-is-the-difference/](https://www.ranktracker.com/ro/blog/static-vs-dynamic-linking-%20%20%20%20%20what-is-the-difference/)