

Colegiul Național „Alexandru Lahovari”  
Râmnicu Vâlcea

# Lucrare pentru obținerea atestatului profesional

**Profesor coordonator: Pădurețu Emil**

Elev: Dobrete Andrei–Robert  
Clasa a XII-a B

Mai, 2022

# Calculator

# Cuprins

<b>1</b>	<b>Despre proiect</b>	<b>2</b>
1.1	Opțiuni . . . . .	2
1.2	De ce am ales acest proiect? . . . . .	2
1.3	Interfață . . . . .	2
<b>2</b>	<b>Limbajul de programare C++</b>	<b>3</b>
2.1	Introducere . . . . .	3
2.2	De ce am ales C++? . . . . .	3
<b>3</b>	<b>Librăria Qt</b>	<b>4</b>
3.1	Introducere . . . . .	4
3.2	Qt Creator . . . . .	4
3.3	De ce am ales Qt? . . . . .	4
<b>4</b>	<b>Instrucțiuni pentru compilare</b>	<b>5</b>
4.1	Pentru Linux . . . . .	5
4.2	Pentru Windows/macOS/altele . . . . .	5
<b>5</b>	<b>Codul sursă</b>	<b>6</b>
5.1	main.cpp . . . . .	6
5.2	calculator.h . . . . .	6
5.3	calculator.cpp . . . . .	7
5.4	calculator.ui . . . . .	12
<b>6</b>	<b>Programe utilizate</b>	<b>13</b>
6.1	Pentru scrierea codului . . . . .	13
6.2	Pentru scrierea documentației . . . . .	13
<b>7</b>	<b>Bibliografie</b>	<b>14</b>

# 1 Despre proiect

Un calculator software este un calculator care a fost implementat ca un program, în loc să fie un dispozitiv hardware fizic dedicat acestui scop.

## 1.1 Opțiuni

- operațiile de bază:
  - adunare (+)
  - scădere (-)
  - înmulțire ( $\cdot$ )
  - împărțire ( $/$ )
- funcția exponențială/ridicarea la putere ( $x^y$ )
- radicalul/rădăcina pătrată a numărului ( $\sqrt{\phantom{x}}$ )
- schimbarea semnului (+/-)
- inversul numărului ( $\frac{1}{x}$ )
- procente (%)
- funcționalitate de memorie (MR/MC/M+/M-)

## 1.2 De ce am ales acest proiect?

Motivul pentru care am ales să fac un calculator pentru proiectul de atestat este deoarece calculatorul este unul din cele mai simple, des folosite, dar și indispensabile programe pe care cineva le folosește atunci când utilizează un sistem de calcul.

## 1.3 Interfață

da

## 2 Limbajul de programare C++

### 2.1 Introducere

C++ este un limbaj de programare general creat de către informaticianul Bjarne Stroustrup de la Bell Labs în anul 1985 ca o extensie a limbajului de programare C, numit inițial „C cu clase”.

Limbajul s-a maturizat destul de mult în timp, iar C++ are acum programare orientată pe obiecte și caracteristici mai avansate, precum manipulare low-level a memoriei.

Este un limbaj de programare compilat și există multe compilatoare create de diferite companii/organizații, precum Fundația Software-ului Liber (GNU GCC), LLVM (Clang), Microsoft (Visual C++), Intel (icx), Oracle (Oracle Developer Studio C++) și IBM (IBM XL C++).

C++ este standardizat de ISO<sup>1</sup>, cea mai recentă versiune a standardului fiind publicată în decembrie 2020, numită C++20.

C++ a fost dezvoltat 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 și excepții. Limbajul a fost standardizat în anul 1998 ca și ISO 14882:1998<sup>2</sup>.

În anii 1990, C++ a devenit unul dintre cele mai populare limbaje de programare comerciale, rămânând astfel până în ziua de astăzi.

### 2.2 De ce am ales C++?

Motivele pentru care am ales să îmi scriu proiectul de atestat în limbajul C++ sunt următoarele:

- am experiență anterioară cu acesta
- este un limbaj de programare folosit în multe domenii
- chiar dacă librăria grafică în care am scris programul este valabilă și pentru alte limbaje de programare, precum Python, C# sau Rust, Qt are cel mai mult suport pentru C++

---

<sup>1</sup>engleză: International Organization for Standardization (Organizația Internațională de Standardizare)

<sup>2</sup>[www.iso.org/standard/25845.html](http://www.iso.org/standard/25845.html)

## 3 Librăria Qt

### 3.1 Introducere

Qt este un framework cross-platform, creat cu scopul de a fi o librărie utilizată pentru crearea interfețelor grafice (GUI<sup>3</sup>), precum și pentru crearea de programe care rulează pe diferite platforme software ca și Linux, Windows, macOS, Android, iOS sau sisteme embedded, fără a fi necesare schimbări în cod, de cele mai multe ori, în timp ce programul scris este un program nativ platformei, cu capabilități și viteză native.

Qt este dezvoltat de Compania Qt și de Proiectul Qt, sub guvernanta open-source. Qt este valabil sub o licență comercială, dar și sub o licență open-source (GPL și LGPL)<sup>4</sup>.

Cele mai cunoscute utilizări ale Qt sunt mediul de desktop KDE Plasma pentru sistemele de operare Unix-like<sup>5</sup> (ex.: Linux/BSD), precum și programele VirtualBox, Google Earth, Autodesk Maya, Autodesk 3ds Max, DaVinci Resolve, OBS, VLC, Wireshark, qBittorrent.

### 3.2 Qt Creator

Qt Creator este un IDE<sup>6</sup> cross-platform pentru C++, JavaScript și QML care simplifică crearea programelor cu o interfață grafică. Este parte din SDK<sup>7</sup>-ul pentru framework-ul Qt pentru crearea aplicațiilor GUI. Include un debugger vizual și un designer de interfețe grafice WYSIWYG<sup>8</sup>

Editorul acestuia include caracteristici comune în alte IDE-uri, precum evidențierea sintaxei prin culori, auto-completare și mesaje despre erori în cod de la servere LSP.

Qt Creator poate folosi compilatorul GNU GCC, MinGW, Visual C++ sau Clang.

### 3.3 De ce am ales Qt?

Motivele pentru care am ales să mă folosesc de librăria Qt pentru a-mi scrie proiectul de atestat sunt următoarele:

- folosesc zilnic, de cel puțin trei ani, unul din cele mai cunoscute proiecte în care se utilizează Qt, mediul de desktop KDE Plasma și suita de programe KDE, lucru care m-a convins că este un framework robust și de încredere
- am vrut să învăț ceva nou
- am considerat că învățarea folosirii unui framework pentru crearea programelor cu interfață grafică al cărui proiecte se pot compila, mai târziu, pe aproape toate platformele software importante în ziua de astăzi este ceva care va îmi va fi util în viitor

---

<sup>3</sup>engleză: graphical user interface (interfață grafică)

<sup>4</sup>GNU General Public License și GNU Lesser General Public License: [www.gnu.org/licenses](http://www.gnu.org/licenses)

<sup>5</sup>[en.wikipedia.org/wiki/Unix-like](http://en.wikipedia.org/wiki/Unix-like)

<sup>6</sup>engleză: integrated development environment (mediu integrat de dezvoltare)

<sup>7</sup>engleză: software development kit (trusă de dezvoltare a programelor)

<sup>8</sup>engleză: what you see is what you get (ceea ce vezi este ceea ce primești)

## 4 Instrucțiuni pentru compilare

Personal, am compilat acest program doar pe Linux, așa că instrucțiunile mele vor fi puțin neclare pentru alte sisteme de operare.

Prima dată, trebuie să instalați Qt Creator:

### 4.1 Pentru Linux

Pe Linux, tot ce trebuie să faceți este să instalați Qt Creator cu ajutorul managerului de pachete al distribuției dumneavoastră, sau cu ajutorul Flatpak:

- bazate pe Debian (ex: Ubuntu, Linux Mint, Elementary OS, Pop\_OS etc.):  
`$ sudo apt install qtcreator`
- bazate pe RHEL (ex: Fedora, CentOS, Rocky Linux etc.):  
`$ sudo dnf install qt-creator`
- bazate pe Arch (ex: Manjaro, EndeavourOS, Arco Linux etc.):  
`$ sudo pacman -Syu qtcreator`
- cu Flatpak (nu contează distribuția):  
`$ sudo flatpak install io.qt.QtCreator`

### 4.2 Pentru Windows/macOS/altele

Pentru restul sistemelor de operare, sau în cazul (extrem de rar) în care nu puteți găsi pachetul pentru distribuția de Linux pe care o folosiți, puteți intra pe linkul [www.qt.io/download-qt-installer](http://www.qt.io/download-qt-installer) și să descărcați installer-ul de acolo. Dar, din încercările mele, am observat că acel instalator necesită un cont pe site-ul Qt.

După ce ați instalat Qt Creator, puteți deschide direct proiectul inclus cu CD-ul care vine cu documentația, sau să copiați codul din acest document într-un proiect nou.

## 5 Codul sursă

### 5.1 main.cpp

Acesta este fișierul care conține funcția principală a programului. Nu sunt multe modificări aici.

```
1  #include "calculator.h"
2  #include <QApplication>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication a(argc, argv);
7      Calculator w;
8      w.show();
9      return a.exec();
10 }
```

### 5.2 calculator.h

Acesta este fișierul header, în care sunt declarate clasele, precum și funcțiile care vor urma să fie declarate.

```
1  #ifndef CALCULATOR_H
2  #define CALCULATOR_H
3  #include <QMainWindow>
4
5  QT_BEGIN_NAMESPACE
6  namespace Ui
7  {
8      class Calculator;
9  }
10 QT_END_NAMESPACE
11
12 class Calculator : public QMainWindow
13 {
14     Q_OBJECT
15
16 public:
17     Calculator(QWidget *parent = nullptr);
18     ~Calculator();
19
20 private:
21     Ui::Calculator *ui;
22
23 private slots:
```



```

24     void NumPressed();
25     void MathButtonPressed();
26     void EqualButtonPressed();
27     void ChSignPressed();
28     void AllClearButtonPressed();
29     void ClearButtonPressed();
30     void MemButtonPressed();
31     void MemResultButtonPressed();
32     void MemClearButtonPressed();
33     void SqrtButtonPressed();
34     void InvButtonPressed();
35     void PercButtonPressed();
36     void PeriodButtonPressed();
37 };
38 #endif // CALCULATOR_H

```

### 5.3 calculator.cpp

Acesta este fișierul cel mai important din proiect. Este fișierul care declară funcțiile necesare, creează variabilele și spune programului ce funcții trebuie folosite la apăsarea căror butoane.

```

1  #include "calculator.h"
2  #include "../ui_calculator.h"
3
4  double calcVal    = 0.0,
5         mrVal      = 0.0;
6  bool divTrigger   = false,
7         multTrigger = false,
8         addTrigger  = false,
9         subTrigger  = false,
10        powTrigger  = false,
11        mrTrigger    = false,
12        mpTrigger    = false,
13        mmTrigger    = false;
14
15 Calculator::Calculator(QWidget *parent):
16     QMainWindow(parent),
17     ui(new Ui::Calculator)
18 {
19     ui->setupUi(this);
20
21     ui->Display->setText(QString::number(calcVal));
22     QPushButton *numButtons[10];
23
24     // Ce functie se executa atunci cand se apasa ce buton

```

```

25     // pentru cifre
26     for(int i=0; i<10; ++i)
27     {
28         QString butName = "Button" + QString::number(i);
29         numButtons[i] = Calculator::findChild<QPushButton *>(butName);
30         connect(numButtons[i], SIGNAL(released()), this, SLOT(NumPressed()));
31     }
32
33     // pentru operatiile de baza
34     connect(ui->ButtonPlus, SIGNAL(released()), this, SLOT(MathButtonPressed()));
35     connect(ui->ButtonMinus, SIGNAL(released()), this, SLOT(MathButtonPressed()));
36     connect(ui->ButtonMultiply, SIGNAL(released()), this,
→    SLOT(MathButtonPressed()));
37     connect(ui->ButtonDivide, SIGNAL(released()), this, SLOT(MathButtonPressed()));
38
39     // pentru operatiile in plus
40     connect(ui->ButtonPow, SIGNAL(released()), this, SLOT(MathButtonPressed()));
41     connect(ui->ButtonSqrt, SIGNAL(released()), this, SLOT(SqrtButtonPressed()));
42     connect(ui->ButtonInverse, SIGNAL(released()), this, SLOT(InvButtonPressed()));
43     connect(ui->ButtonPercent, SIGNAL(released()), this, SLOT(PercButtonPressed()));
44
45     // pentru alte butoane
46     connect(ui->ButtonEquals, SIGNAL(released()), this, SLOT(EqualButtonPressed()));
47     connect(ui->ButtonChSign, SIGNAL(released()), this, SLOT(ChSignPressed()));
48     connect(ui->ButtonAllClear, SIGNAL(released()), this,
→    SLOT(AllClearButtonPressed()));
49     connect(ui->ButtonClear, SIGNAL(released()), this, SLOT(ClearButtonPressed()));
50     connect(ui->ButtonPeriod, SIGNAL(released()), this,
→    SLOT(PeriodButtonPressed()));
51
52     // pentru butoanele de memorie
53     connect(ui->ButtonMemResult, SIGNAL(released()), this,
→    SLOT(MemResultButtonPressed()));
54     connect(ui->ButtonMemClear, SIGNAL(released()), this,
→    SLOT(MemClearButtonPressed()));
55     connect(ui->ButtonMemPlus, SIGNAL(released()), this, SLOT(MemButtonPressed()));
56     connect(ui->ButtonMemMinus, SIGNAL(released()), this, SLOT(MemButtonPressed()));
57 }
58
59 Calculator::~Calculator()
60 {
61     delete ui;
62 }
63
64 // Actiunea pentru apasarea cifrelor
65 void Calculator::NumPressed()
66 {

```

```

67     QPushButton *button = (QPushButton *)sender();
68     QString butVal = button->text();
69     QString displayVal = ui->Display->text();
70
71     if((displayVal.toDouble() == 0) || (displayVal.toDouble() == 0.0))
72         ui->Display->setText(butVal);
73     else
74     {
75         QString newVal = displayVal + butVal;
76         double dblNewVal = newVal.toDouble();
77         ui->Display->setText(QString::number(dblNewVal, 'g', 16)); //scrie numarul
78         ↪ in exponenti daca trece de 16 cifre
79     }
80
81     // Actiunea pentru detectarea apasarii semnelor
82     void Calculator::MathButtonPressed()
83     {
84         //reseteaza valorile semnelor
85         addTrigger = false;
86         subTrigger = false;
87         multTrigger = false;
88         divTrigger = false;
89         powTrigger = false;
90
91         QString displayVal = ui->Display->text();
92         calcVal = displayVal.toDouble();
93
94         QPushButton *button = (QPushButton *)sender();
95         QString butVal = button->text();
96
97         if(QString::compare(butVal, "+", Qt::CaseInsensitive) == 0)
98             addTrigger = true;
99         else if(QString::compare(butVal, "-", Qt::CaseInsensitive) == 0)
100             subTrigger = true;
101         else if(QString::compare(butVal, "*", Qt::CaseInsensitive) == 0)
102             multTrigger = true;
103         else if(QString::compare(butVal, "/", Qt::CaseInsensitive) == 0)
104             divTrigger = true;
105         else if(QString::compare(butVal, "^", Qt::CaseInsensitive) == 0)
106             powTrigger = true;
107
108         ui->Display->setText(""); //sterge ecran dupa apasarea semmului
109     }
110
111     // Actiunea pentru apasarea egalului
112     void Calculator::EqualButtonPressed()

```

```

113 {
114     double solution = 0.0;
115     QString displayVal = ui->Display->text();
116     double dblDisplayVal = displayVal.toDouble();
117
118     //calcularea solutiei
119     if(addTrigger || subTrigger || multTrigger || divTrigger || powTrigger)
120     {
121         if(addTrigger) solution = calcVal + dblDisplayVal;
122         else if(subTrigger) solution = calcVal - dblDisplayVal;
123         else if(multTrigger) solution = calcVal * dblDisplayVal;
124         else if(divTrigger) solution = calcVal / dblDisplayVal;
125         else if(powTrigger) solution = qPow(calcVal, dblDisplayVal);
126     }
127     ui->Display->setText(QString::number(solution)); //afisare rezultat
128 }
129
130 // Actiunea pentru schimbarea semnului
131 void Calculator::ChSignPressed()
132 {
133     QString displayVal = ui->Display->text();
134     double dblDisplayVal = displayVal.toDouble();
135     double dblDisplayValSign = -1 * dblDisplayVal;
136     ui->Display->setText(QString::number(dblDisplayValSign)); //arata schimbarea
137     ↪ semnului pe ecran
138 }
139
140 // Actiunea pentru apasarea de clear
141 void Calculator::AllClearButtonPressed()
142 {
143     ui->Display->setText(QString::number(0.0));
144 }
145
146 // Actiunea pentru apasarea butoanelor de memorie
147 void Calculator::MemButtonPressed()
148 {
149     //reseteaza valorile butoanelor
150     mpTrigger = false;
151     mmTrigger = false;
152
153     // mrVal = 0.0;
154
155     QString displayVal = ui->Display->text();
156     double dblDisplayVal = displayVal.toDouble();
157
158     QPushButton *button = (QPushButton *)sender();
159     QString butVal = button->text();

```

```

159
160     if(QString::compare(butVal, "M+", Qt::CaseInsensitive) == 0)
161         mpTrigger = true;
162     else if(QString::compare(butVal, "M-", Qt::CaseInsensitive) == 0)
163         mmTrigger = true;
164
165     if(mpTrigger || mmTrigger)
166     {
167         if(mpTrigger) mrVal = mrVal + dblDisplayVal;
168         else if(mmTrigger) mrVal = mrVal - dblDisplayVal;
169     }
170
171 }
172
173 // Actiunea pentru apasarea rezultatului de memorie
174 void Calculator::MemResultButtonPressed()
175 {
176     ui->Display->setText(QString::number(mrVal)); //afisare rezultat
177     mrVal = 0.0;
178 }
179
180 // Actiunea pentru apasarea clear-ului de memorie
181 void Calculator::MemClearButtonPressed()
182 {
183     mrVal = 0.0;
184 }
185
186 // Actiune pentru apasarea radicalului
187 void Calculator::SqrtButtonPressed()
188 {
189     QString displayVal = ui->Display->text();
190     double dblDisplayVal = displayVal.toDouble();
191     double dblDisplayValSqrt = qSqrt(dblDisplayVal);
192     ui->Display->setText(QString::number(dblDisplayValSqrt));
193 }
194
195 // Actiune pentru apasarea inversului
196 void Calculator::InvButtonPressed()
197 {
198     QString displayVal = ui->Display->text();
199     double dblDisplayVal = displayVal.toDouble();
200     double dblDisplayValInv = 1 / dblDisplayVal;
201     ui->Display->setText(QString::number(dblDisplayValInv));
202 }
203
204 // Actiune pentru apasarea procentului
205 void Calculator::PercButtonPressed()

```

```

206 {
207     QString displayVal = ui->Display->text();
208     double dblDisplayVal = displayVal.toDouble();
209     double dblDisplayValPerc = dblDisplayVal / 100;
210     ui->Display->setText(QString::number(dblDisplayValPerc));
211 }
212
213 // Actiune pentru apasarea virgulei
214 void Calculator::PeriodButtonPressed()
215 {
216     QString displayVal = ui->Display->text();
217     displayVal = displayVal + ".";
218     ui->Display->setText(displayVal);
219 }
220
221 // Actiune pentru apasarea backspace
222 void Calculator::ClearButtonPressed()
223 {
224     QString displayVal = ui->Display->text();
225     if(! displayVal.isEmpty())
226         displayVal.chop(1);
227     else displayVal = "0";
228     ui->Display->setText(displayVal);
229 }

```

## 5.4 calculator.ui

Acesta este fișierul care conține layout-ul grafic al programului (ex.: unde se află butoanele, felul în care arată etc.). Acesta a fost auto-generat de către componenta de ‘Design’ din interiorul IDE-ului Qt Creator (secțiunea 3.2). Este, de asemenea, și cel mai lung fișier din întregul proiect.

## 6 Programe utilizate

### 6.1 Pentru scrierea codului

- Qt Creator (secțiunea 3.2) ([www.qt.io/download-qt-installer](http://www.qt.io/download-qt-installer)) – scrierea codului + design GUI
- Neovim ([neovim.io](http://neovim.io)) cu multe plugin-uri și configurări ([gitlab.com/Andy3153/andy3153-init\\_vim](https://gitlab.com/Andy3153/andy3153-init_vim)) – scrierea codului
- GNU GCC ([gcc.gnu.org](http://gcc.gnu.org)) – compilatorul utilizat

### 6.2 Pentru scrierea documentației

- Neovim ([neovim.io](http://neovim.io)) cu multe plugin-uri și configurări ([gitlab.com/Andy3153/andy3153-init\\_vim](https://gitlab.com/Andy3153/andy3153-init_vim)) – scrierea documentației
- VimTeX ([github.com/lervag/vimtex](https://github.com/lervag/vimtex)) – plugin pentru Vim/Neovim pentru scrierea documentelor  $\text{\TeX}$
- $\text{\LaTeX}$ mk – pentru compilarea mai ușoară a documentelor  $\text{\TeX}$
- $\text{\TeX}$  Live – distribuția de  $\text{\LaTeX}$
- $\text{\XeLaTeX}$  – engine-ul de  $\text{\TeX}$  folosit de mine

## 7 Bibliografie

- Wikipedia: C++ (engleză): [en.wikipedia.org/wiki/C%2B%2B](https://en.wikipedia.org/wiki/C%2B%2B)
- Wikipedia: C++ (română): [ro.wikipedia.org/wiki/C%2B%2B](https://ro.wikipedia.org/wiki/C%2B%2B)
- qt.io: documentație: [doc.qt.io/all-topics.html](https://doc.qt.io/all-topics.html)
- Wikipedia: Qt (engleză): [en.wikipedia.org/wiki/Qt\\_\(software\)](https://en.wikipedia.org/wiki/Qt_(software))
- Wikipedia: Qt (română): [ro.wikipedia.org/wiki/Qt](https://ro.wikipedia.org/wiki/Qt)
- Wikipedia: Qt Creator: [en.wikipedia.org/wiki/Qt\\_Creator](https://en.wikipedia.org/wiki/Qt_Creator)