1. Proiectarea unei aplicaţii de vedere artificială

Librăria de vedere artificială Open Computer Vision (OpenCV)
Configurarea unui proiect bazat pe OpenCV
Citirea şi afişarea imaginilor

Scopul acestui laborator este acela de a se introduce principalele elementele ale librăriei OpenCV, dintre acestea amintind: citirea, afişarea sau salvarea imaginilor. Primii paşi în dezvoltarea unei aplicaţii care are la bază librăria amintită sunt aceia de instalare a librăriei şi de configurare a unui proiect într-un mediu de dezvoltare software (IDE) (de ex. MS Visual Studio, Eclipse). Pe parcursul laboratoarelor prezentate în acest îndrumar se va folosi sistemul *Microsoft Visual C++ 2010*.

1.1 Instalarea librăriei OpenCV

OpenCV este o librărie open source destinată dezvoltării aplicaţiilor de vedere artificială. Datorită licenţei BSD, librăria poate fi utilizată atât pentru aplicaţii academice, cât şi pentru sisteme comerciale. Ultimele versiuni ale librăriei se găsesc la adresa `www.opencv.org`.

Pagina de Download a site-ului librăriei pune la dispoziţia utilizatorilor diferite versiuni de OpenCV, specifice diferitelor platforme de operare, precum Linux, Unix, Mac, iOS, sau Windows. Arhiva descărcată conţine, de obicei, un folder denumit în funcţie de versiunea librăriei, spre exemplu OpenCV 2.43, în interiorul căruia se găsesc o serie de directoare, prezentate în Fig. 1.1. Dintre acestea, amintim directorul `include`, în care se află fişierele care vor fi incluse în proiecte; directorul `modules` care conţine fişierele sursă ale librăriei; directorul `doc` în care se regăseşte documentaţia specifică librăriei; directorul `samples` în care se găsesc exemple de programe pentru lucrul cu librăria.

În situaţia în care este utilizat sistemul de operare Windows, împreună cu mediul de programare Visual Studio C++, există posibilitatea de a se descărca un program executabil de instalare automată a librăriei OpenCV.

Începând cu versiunea 2.2, librăria OpenCV este divizată în mai multe module, compilate ca librării separate, în directorul `lib`. Aceste module sunt:

- `opencv_core`, care conţine funcţionalităţile de bază ale librăriei, mai exact structurile de date şi funcţiile aritmetice;
- `opencv_imgproc`, în care sunt stocate principalele funcţii de procesare a imaginilor;
- `opencv_highgui`, utilizat pentru citirea şi salvarea imaginilor, cât şi a fişierelor video, împreună cu o serie de funcţii destinate creării de interfeţe cu utilizatorul;
- `opencv_features2d`, conţine detectoarele şi descriptoarele de puncte cheie, precum şi metodele de potrivire [1] dintre aceste puncte;

[1] Eng. *Matching*

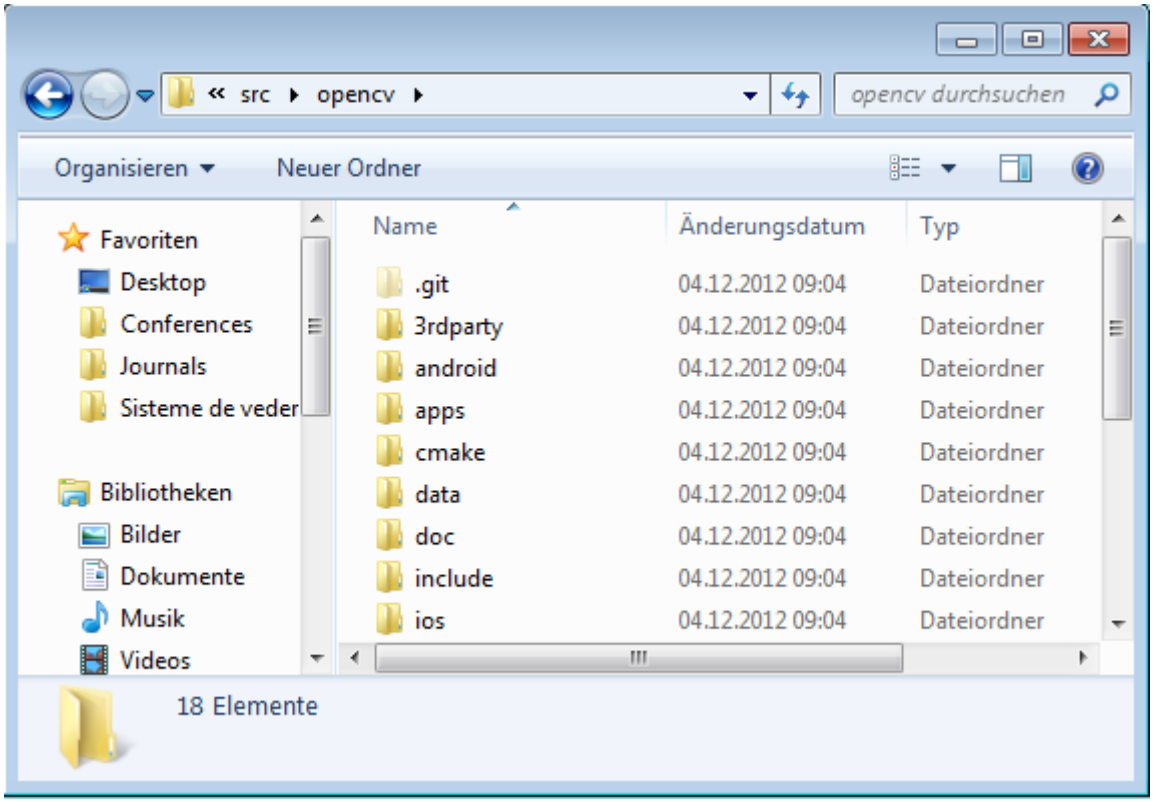

Fig. 1.1 Structura de directoare a librăriei OpenCV.

- `opencv_calib3d`, dezvoltat pentru calibrarea camerelor şi estimarea geometriei scenelor, prin utilizarea de camere stereo;
- `opencv_video`, utilizat pentru estimarea mişcării şi urmărirea formelor;
- `opencv_objdetect`, care conţine funcţii pentru detectarea obiectelor, precum şi recunoaşterea persoanelor, respectiv a feţelor;
- `opencv_ml`, în acest modul se găsesc algoritmii de inteligenţă artificială;
- `opencv_flann`, este utilizat pentru implementarea metodelor de calcul geometric (ex. determinarea celui mai apropiat vecin în spaţiul 3D);
- `opencv_gpu`, necesar stocării funcţiilor implementate pe structuri de calcul paralel [2];
- `opencv_legacy`, utilizat pentru a se realiza o compatibilizare a noilor versiuni cu vechile variante ale librăriei OpenCV.

Fiecare dintre modulele menţionate anterior au câte un fişier header asociat, localizat în directorul `include`. În mod tipic, un program în OpenCV va începe prin includerea modulelor necesare aplicaţiei, spre exemplu:

```
1  #include <opencv2/core/core.hpp>
2  #include <opencv2/imgproc/imgproc.hpp>
3  #include <opencv2/highgui/highgui.hpp>
```

Un program OpenCV ce începe cu header-ul:

```
1  #include "cv.h"
```

reprezintă o aplicaţie dezvoltată cu o versiune veche a librăriei.

Versiunea cea mai recentă a codului librăriei OpenCV poate fi descărcată, utilizându-se sistemul de management al codului sursă GIT, de la adresa `git://code.opencv.org/opencv.git`

[2] Eng. *Graphical Processing Unit*

1.2 Mediul de programare MS Visual C++

În sistemul de operare Windows se pot crea uşor aplicaţii care folosesc librăria OpenCV, prin utilizarea mediului de programare MS Visual C++. Se pot crea atât aplicaţii simple, de consolă, cât şi aplicaţii care utilizează interfeţe cu utilizatorul (GUI). De-a lungul acestui îndrumar vor fi create doar aplicaţii tip consolă, în mediul Visual Studio 2010. Cu toate acestea, aceleaşi principii se pot aplica oricărei versiuni a mediului MS Visual Studio.

La prima lansare a programului Visual Studio, se poate seta mediul de programare default ca fiind C++. Astfel, Visual Studio va porni de fiecare dată în modul C++. Vom considera ca aţi instalat librăria OpenCV în directorul `C:\OpenCV2.4`, aşa cum s-a explicat în secţiunea anterioară.

În Visual Studio este important să se înţeleagă diferenţa dintre o soluţie şi un proiect. O soluţie este compusă din mai multe proiecte. Spre exemplu, un proiect este un modul software distinct (program sau librărie). Într-o soluţie, diferite proiecte pot împărţii fişiere şi librării. De obicei se crează un director principal pentru soluţie, care conţine directoarele fiecărui proiect.

De asemenea, un proiect Visual C++ se poate compila şi executa în două configuraţii: `Debug` şi `Release`. Modul Debug este utilizat pentru analiza erorilor din codul sursă sau determinarea scăpărilor de memorie. Cu toate astea, modul Debug generează programe care se execută într-un interval de timp mai mare. Astfel, după ce aplicaţia a fost testată, ea poate fi compilată în versiunea Release, versiune ce va fi distribuită ulterior utilizatorilor. Modurile Debug şi Release nu sunt specifice doar mediului MS Visual C++, ci majorităţii sistemelor de dezvoltare a aplicaţiilor software.

1.3 Configurarea unui proiect în MS Visual C++

Un proiect nou în MS Visual C++ se crează utilizând opţiunea `File|New Project` din meniul utilizator. Se va selecta opţiunea `Win32 Console Application`, aşa cum este ilustrat în Fig. 1.2.

În acest moment se specifică locaţia unde se va salva proiectul, cât şi numele său. Există şi opţiunea de a se crea un director pentru proiect în soluţie. Apăsând OK, se vor afişa setările, cum poate fi văzut în Fig. 1.3. Se va selecta un proiect gol (empty project).

Opţiunea specifică MS Visual Studio `Precompiled header` trebuie să fie deselectată. Această opţiune face procesul de compilare mai rapid. Deoarece se doreşte utilizarea standardului ANSI C++, nu vom utiliza headere precompilate. Proiectul va fi creat apăsând butonul `Finish`, urmând să adăugăm codul sursă într-un fişier `main.cpp`.

Pentru a se putea compila şi rula aplicaţii utilizând librăria OpenCV, este necesară setarea căii mediului Visual C++ către locaţia unde se găsesc librăriile OpenCV şi fişierele de tip `include`. Deoarece se vor crea o serie de proiecte care au la bază OpenCV, cea mai bună opţiune de configurare este aceea de a se crea o foaie de proprietăţi [3] care va putea fi reutilizată în fiecare proiect. Acest lucru este posibil prin utilizarea managerului de proprietăţi (`Property Manager`), prezentat în Fig. 1.4. Dacă nu este vizibil, el poate fi accesat cu ajutorul meniului `View`.

În Visual Studio C++ 2010, o foaie de proprietăţi este un fişier de tip XML care descrie setările unui proiect. După cum se poate vedea în Fig. 1.5, acest fişier se poate crea apăsând click-dreapta pe nodul `Debug|Win32` din proiect şi selectându-se opţiunea `Add New Project Property Sheet`.

Noua foaie de proprietăţi este adăugată odată ce este apăsat butonul `Add`, urmând ca ea să fie editată ulterior. Următorul pas este acela de a se realiza un dublu-click pe numele

[3] Eng. *Property Sheet*

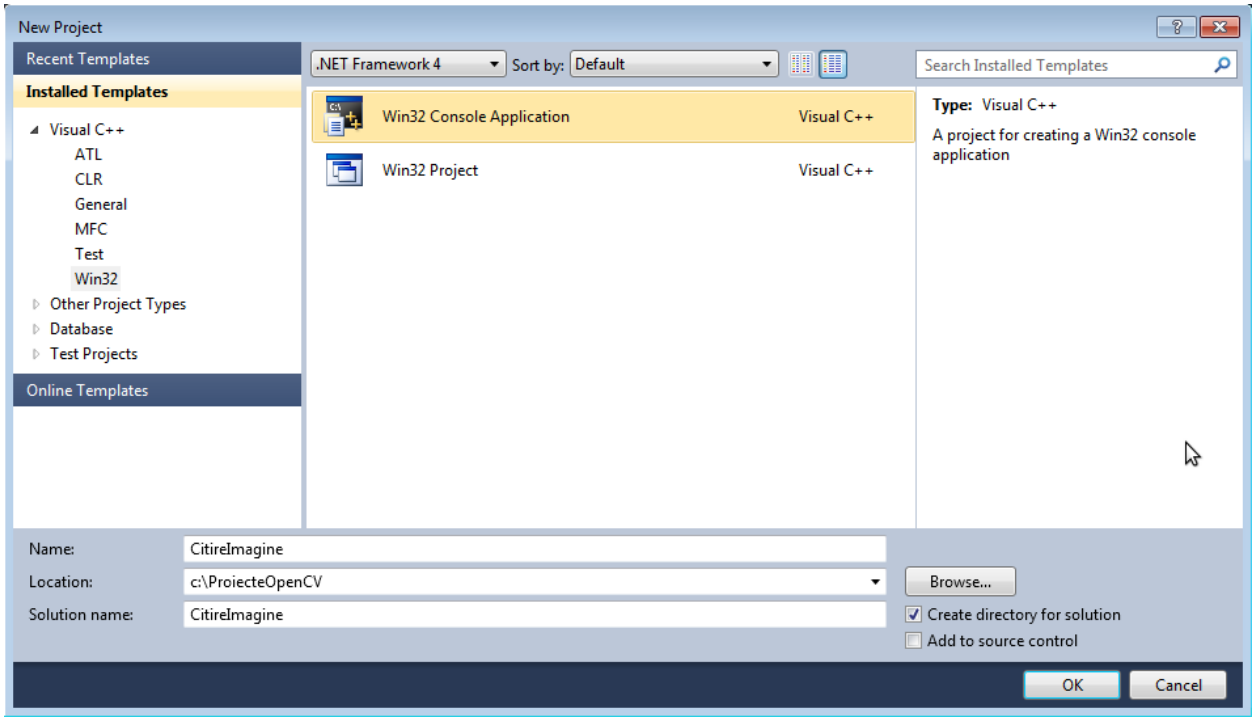

Fig. 1.2 Crearea unui proiect nou în MS Visual Studio C++ 2010.

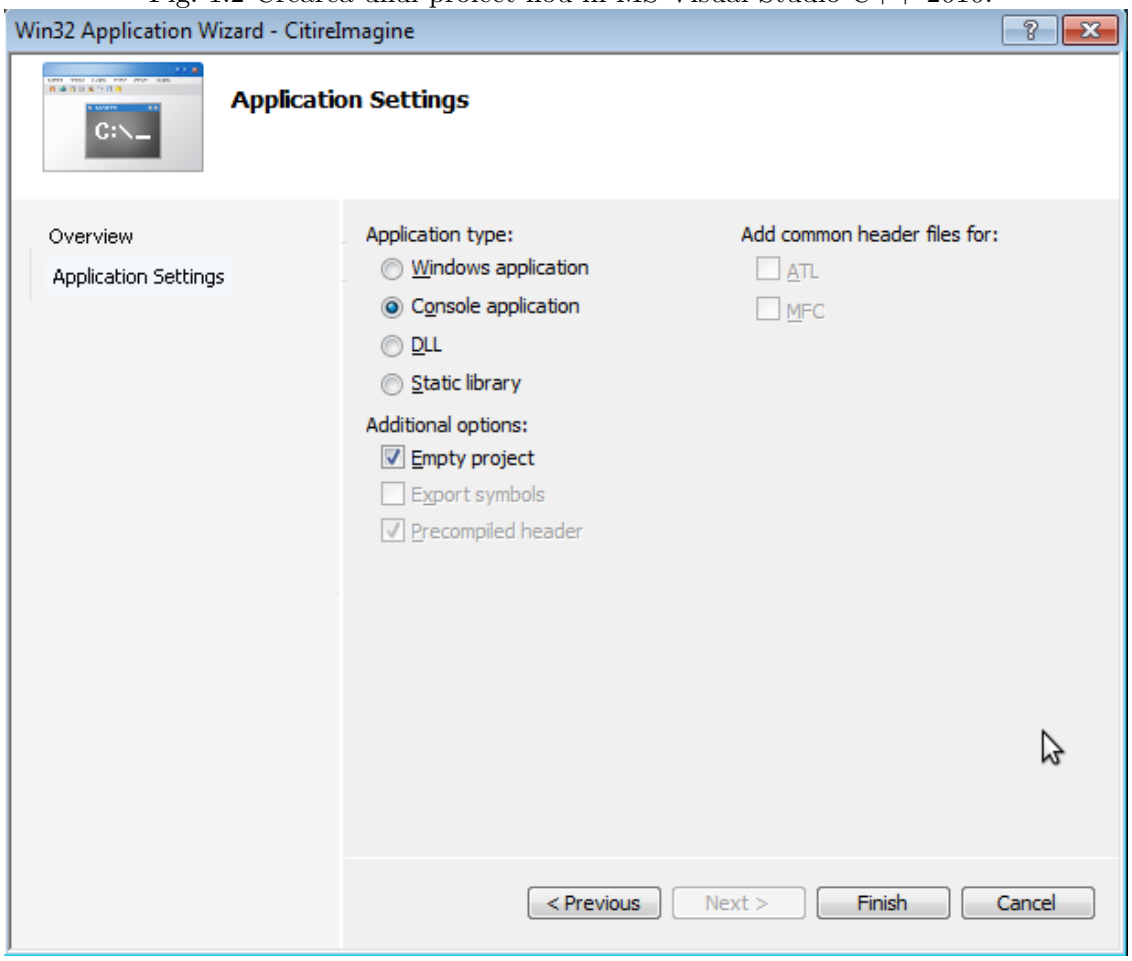

Fig. 1.3 Setările unui nou proiect MS Visual C++ 2010.

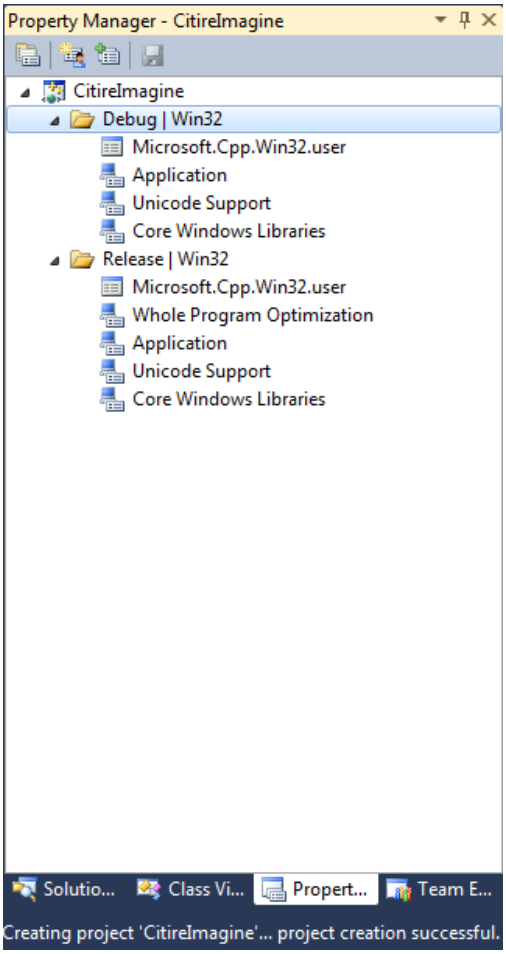

Fig. 1.4 Managerul de proprietăţi din MS Visual Studio C++ 2010.

foii de proprietăţi şi a se selecta opţiunea `VC++ Directories`, după cum este exemplificat în Fig. 1.6.

Se editează câmpul `Include Directories`, accesibil din nodul `VC++ Directories` (v. Fig. 1.7), prin adăugarea căilor către fişierele include din librăria OpenCV (v. Fig. 1.8).

Este necesară efectuarea aceleiaşi operaţii şi pentru câmpul `Library Directories`, prin adăugarea căilor către fişierele de tip librărie din OpenCV (v. Fig. 1.9).

Este important de remarcat faptul că în foaia de proprietăţi setarea locaţiilor fişierelor OpenCV a fost realizată prin căi explicite. În general, este cel mai indicat să se utilizeze variabile de sistem [4] pentru a se desemna locaţia librăriei OpenCV. În acest fel, în cazul în care se doreşte schimbarea versiunii librăriei (spre exemplu de la OpenCV 2.4 la OpenCV 2.5), nu trebuie decât să setaţi variabilele care pointează către locaţia dorită. De asemenea, în cazul dezvoltării de software în echipă, diferiţi utilizatori îşi pot instala OpenCV în locaţii diferite. Utilizându-se o variabilă de sistem, este evitată editarea foii de proprietăţi de către fiecare utilizator în parte. În consecinţă, dacă este definită o variabila de sistem `OPENCV_DIR` care pointează către locaţia `C:\OpenCV 2.4`, atunci cele două directoare care trebuiesc specificate în foaia de proprietăţi vor fi `$(OPENCV_DIR)\include` şi `$(OPENCV_DIR)\lib`.

Următorul pas este de a se specifica care fişiere ale librăriei OpenCV vor fi necesare link-editării, împreună cu codul sursă, pentru a se putea crea aplicaţia executabilă. Astfel, vor fi necesare diferite module din OpenCV, în funcţie de aplicaţia de vedere artificială care urmează a fi creată. Deoarece se doreşte utilizarea aceleiaşi foi de proprietăţi pentru toate

[4] Eng. *Environment Variables*

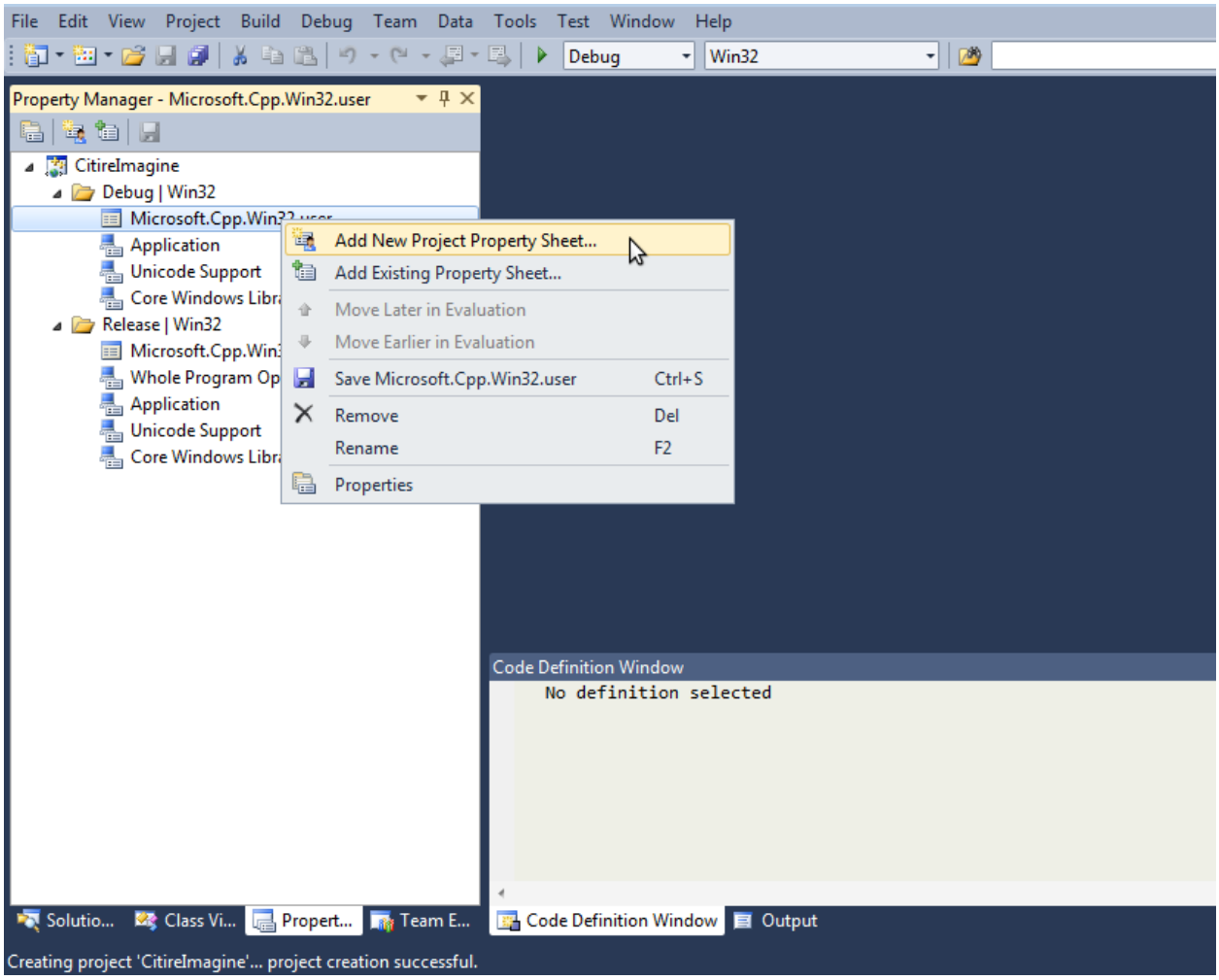

Fig. 1.5 Selectarea opţiunii de adăugare a unei noi foi de proprietăţi.

proiectele, se vor adăuga acele librarii care vor fi utilizate de-a lungul lucrărilor din acest îndrumar. După cum se poate vedea în Fig. 1.10, acest lucru este posibil din opţiunea `Input` a nodului `Linker`.

Este necesară editarea câmpului `Additional Dependencies` (v. Fig. 1.11), prin adăugarea modulelor `opencv_core`, `opencv_imgproc`, `opencv_highgui`, `opencv_features2d` şi `opencv_calib3d`.

Se poate observa că la finalul numelor librăriilor este adăugată litera "d". Acestea reprezintă librăriile pentru modul `Debug`. În situaţia în care este utilizat modul `Release`, este necesară crearea unei noi foi de proprietăţi, aproape identică cu aceea de `Debug` (este utilizată aceeaşi procedură), dar adăugată la nodul `Release|Win32`. În modul `Release` numele librăriilor nu se mai termină cu litera "d".

În acest moment au fost introduse toate noţiunile necesare pentru a se crea, compila şi rula prima aplicaţie de vedere artificială. Adăugarea unui nou fişier sursă poate fi realizată prin utilizarea modulului `Solution Explorer` şi efectuarea unui click-dreapta pe nodul `Source Files`. Este selectată opţiunea `Add New Item`, unde se specifică noul fişier sursă C++, cu numele `main.cpp`, aşa cum este ilustrat în Fig. 1.12. Un nou fişier poate fi, de asemenea, adăugat prin alegerea opţiunii `File|New|File`.

În Fig. 1.13 este prezentat codul sursă al unei aplicaţii care încarcă şi afişează o imagine de pe disc. Rezultatul execuţiei programului este afişat în Fig. 1.14.

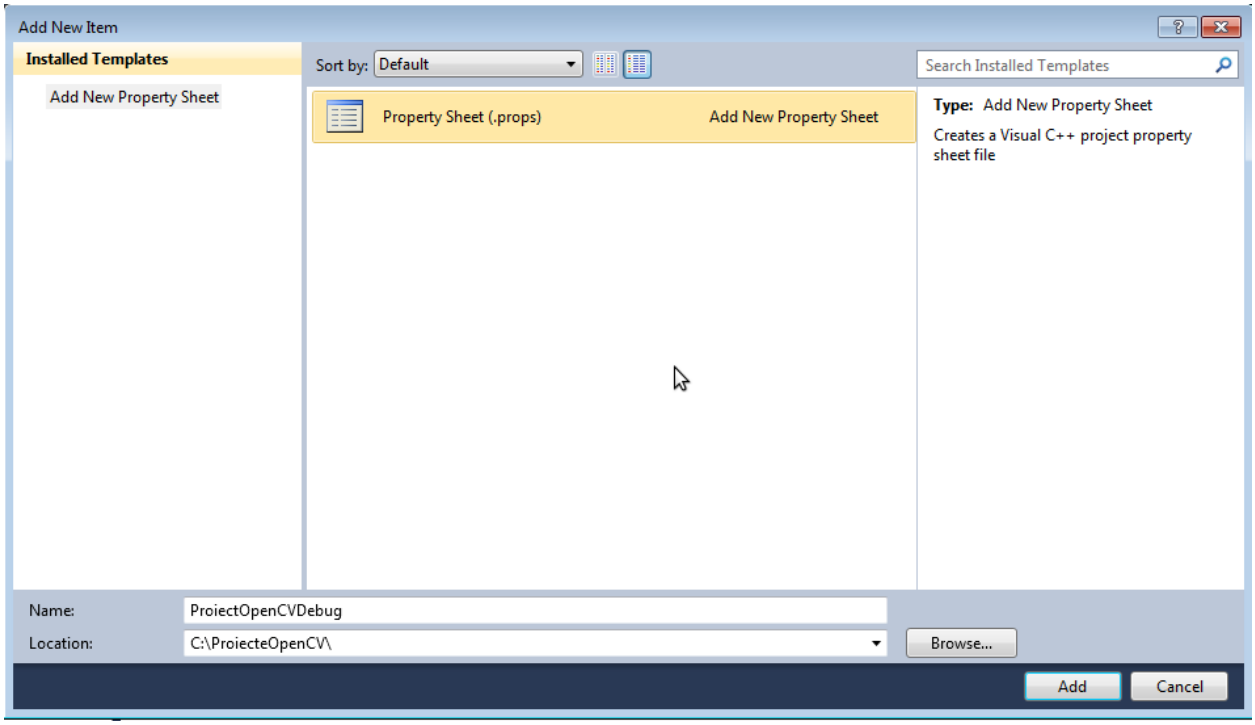

Fig. 1.6 Adăugarea unei noi foi de proprietăţi.

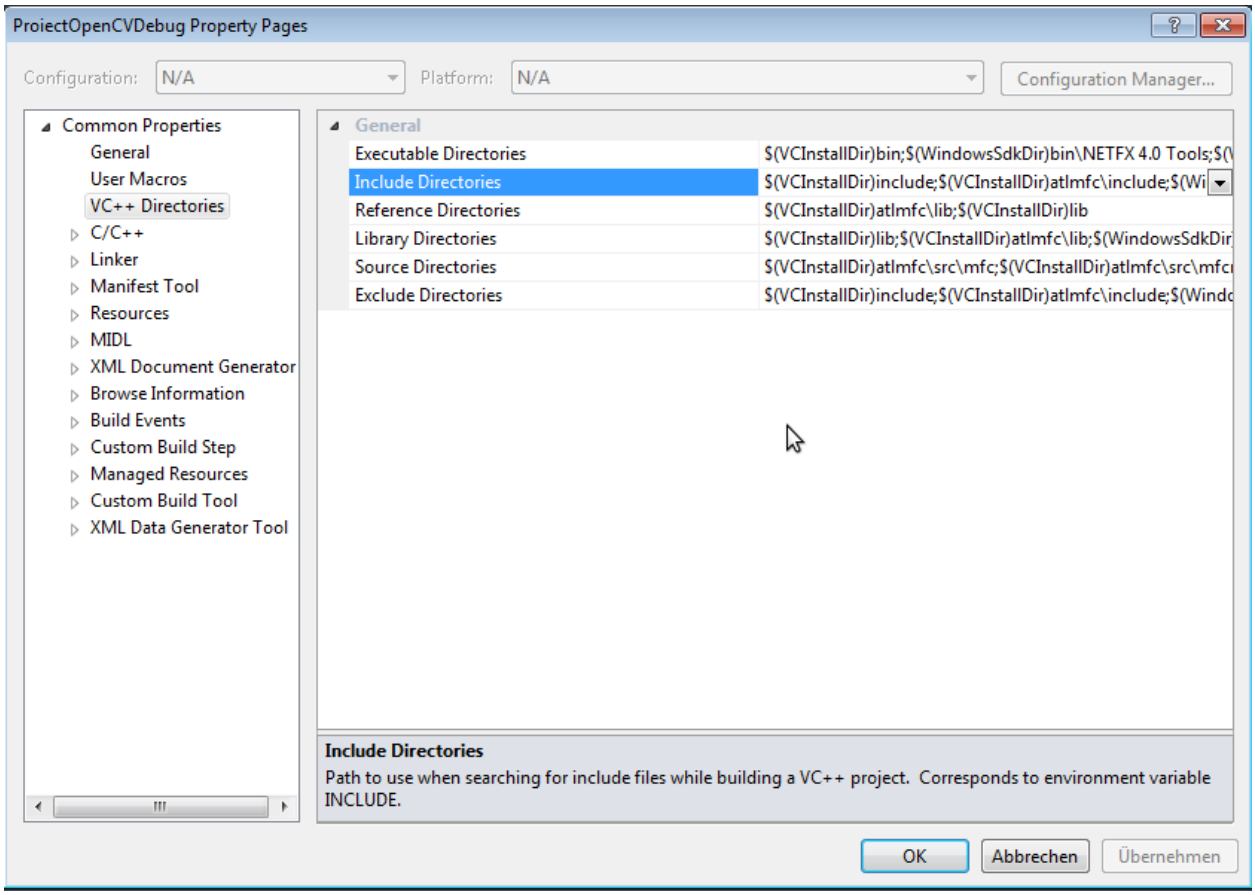

Fig. 1.7 Nodul `VC++ Directories`.

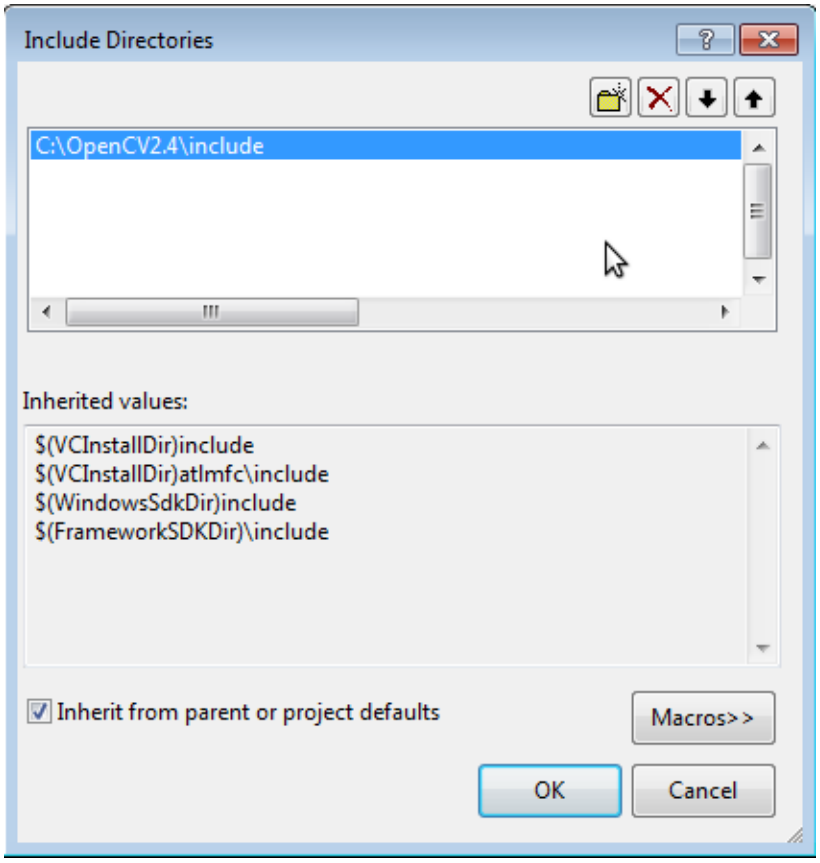

Fig. 1.8 Adăugarea căilor către fişierele include din libraria OpenCV.

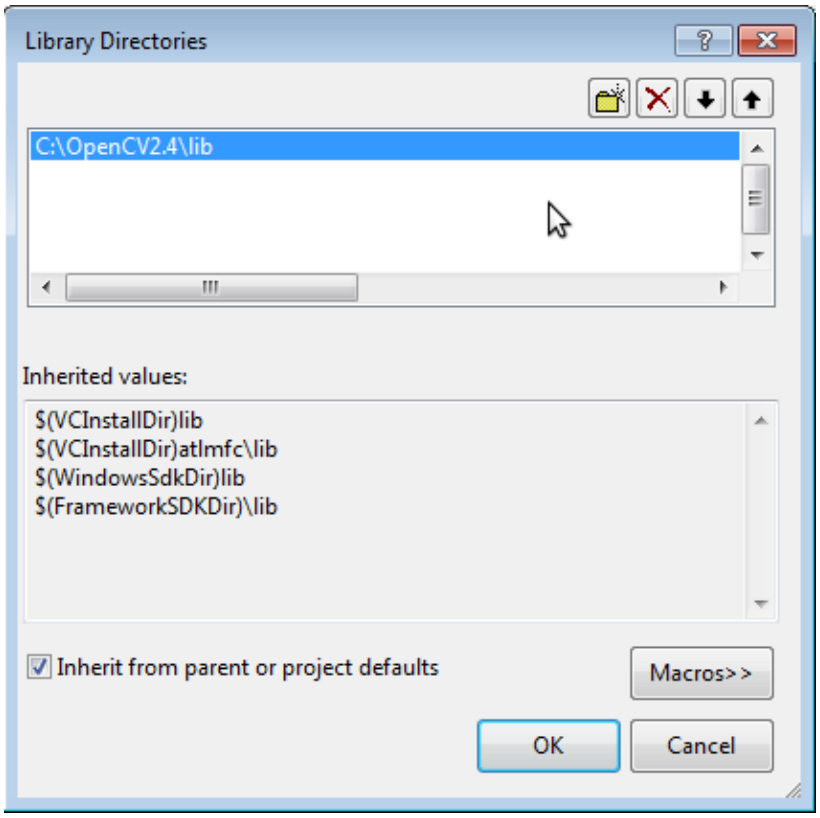

Fig. 1.9 Adăugarea căilor către fişierele librărie din OpenCV.

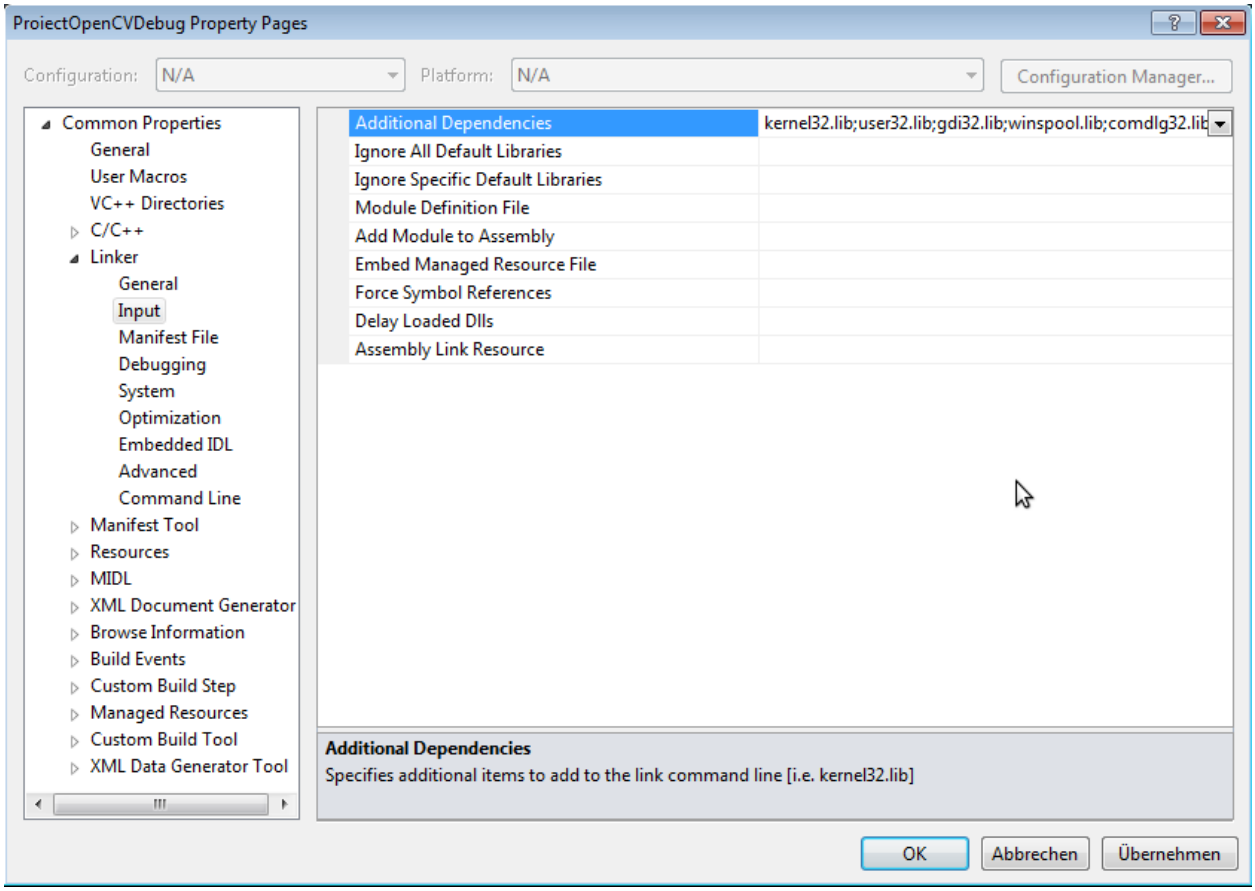

Fig. 1.10 Selectarea optiunii `Input` al nodului `Linker`.

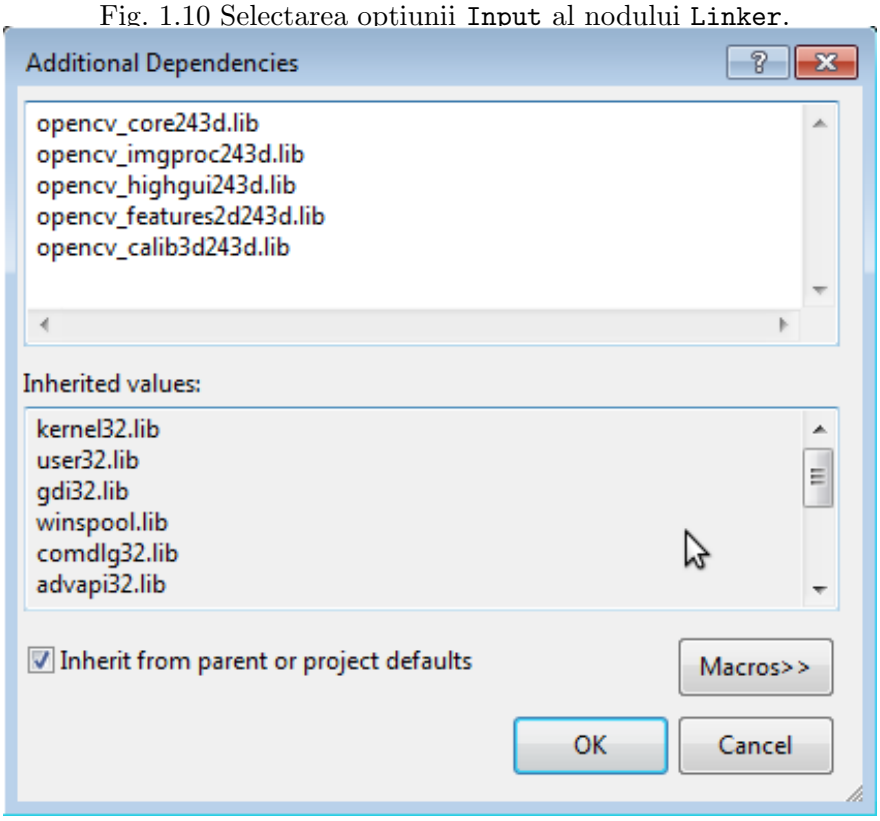

Fig. 1.11 Adăugarea modulelor OpenCV necesare lucrărilor din prezentul îndrumar.

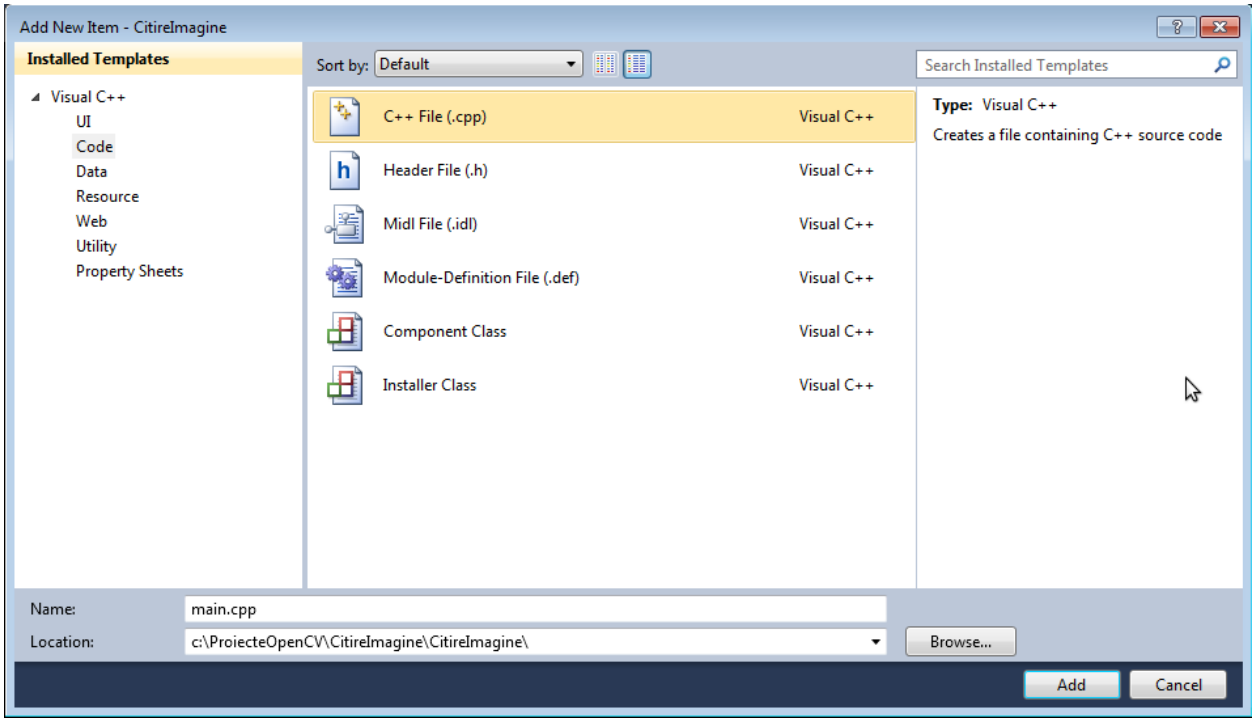

Fig. 1.12 Adăugarea fişierului sursă `main.cpp`.

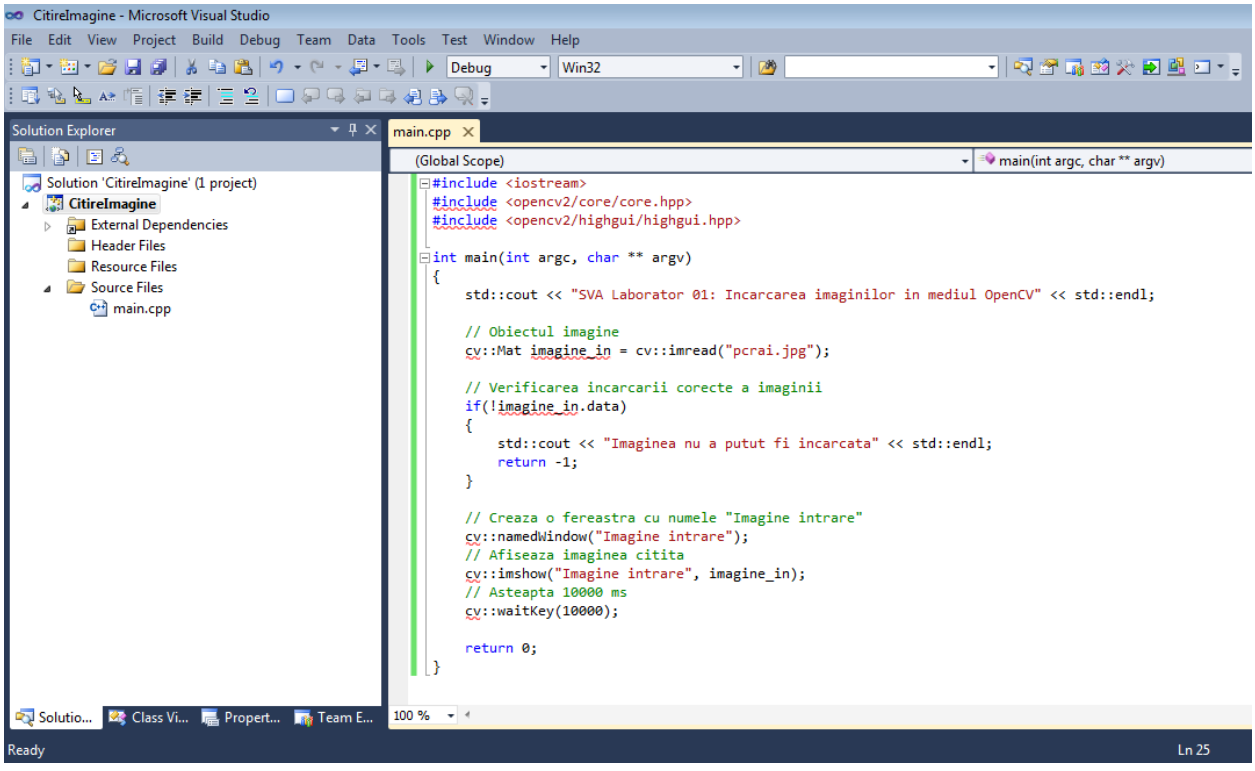

Fig. 1.13 Editarea fişierului `main.cpp` prin adăugarea codului sursă.

1.4 Cerinţe

Să se realizeze citirea şi afişarea unei imagini de pe HDD.

Fig. 1.14 Rezultatul executării aplicaţiei având codul sursă listat în Fig 1.13.

1.5 Codul sursă pentru încărcarea şi afişarea unei imagini

```
1  #include <iostream>
2  #include <opencv2/core/core.hpp>
3  #include <opencv2/highgui/highgui.hpp>
4
5  using namespace cv;
6  using namespace std;
7
8  int main(int argc, char ** argv)
9  {
10   cout << "SVA Laborator 01: Incarcarea imaginilor in mediul OpenCV" ↩
        << endl;
11
12   // Obiectul imagine
13   Mat imagine_in = cv::imread("pcrai.jpg");
14
15   // Verificarea incarcarii corecte a imaginii
16   if(!imagine_in.data)
17   {
18     cout << "Imaginea nu a putut fi incarcata" << endl;
19     return -1;
20   }
21
22   // Creaza o fereastra cu numele "Imagine intrare"
23   namedWindow("Imagine intrare");
24   // Afiseaza imaginea citita
25   imshow("Imagine intrare", imagine_in);
26   // Asteapta 10000 ms
27   waitKey(10000);
28
```

```
29   return 0;
30 }
```

1.6 Descrierea funcţiilor principale

```
2 #include <opencv2/core/core.hpp>
3 #include <opencv2/highgui/highgui.hpp>
```

Declararea fişierelor de tip `include` OpenCV.

```
12 Mat imread(const string& filename, int flags=1)
```

Încarcă o imagine de pe disc.

- `filename`: numele fişierului imagine;
- `flags`: indicator de încărcare a tipului de imagine:
  - `CV_LOAD_IMAGE_ANYDEPTH`: încarcă o imagine pe 16-biţi \32-biţi atunci când imaginea corespunde acestor rezoluţii, altfel converteşte imaginea la 8-biţi;
  - `CV_LOAD_IMAGE_COLOR`: întotdeauna încarcă imaginea color;
  - `CV_LOAD_IMAGE_GRAYSCALE`: converteşte imaginea în niveluri de gri;
  - < 0: încarcă imaginea aşa cum este.

```
22 void namedWindow(const string& winname, int flags=WINDOW_AUTOSIZE)
```

Crează o fereastră.

- `winname`: numele ferestrei;
- `flags`: indicator a tipului de fereastră:
  - `CV_WINDOW_AUTOSIZE`: dimensiunea ferestrei este setată în funcţie de dimensiunea imaginii.

```
24 void imshow(const string& winname, InputArray mat)
```

Afişează o imagine într-o fereastră dată:

- `winname`: numele ferestrei;
- `mat`: imaginea de intrare.

```
26 int waitKey(int delay=0)
```

Aşteaptă ca o tastă să fie apăsată:

- `delay`: întârzierea în milisecunde (0 reprezintă aşteptare infinită).