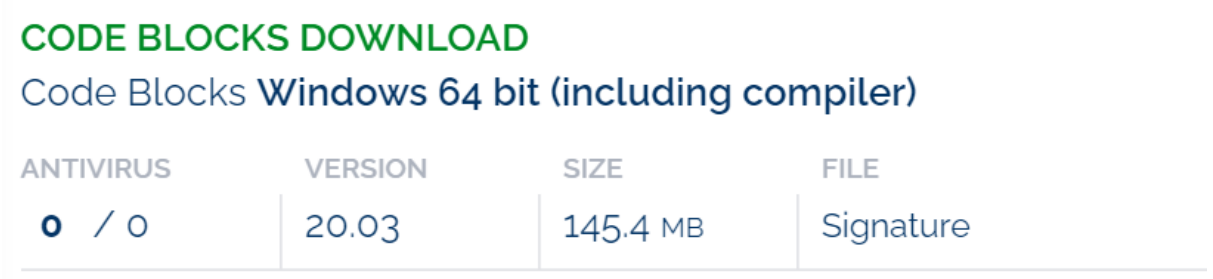
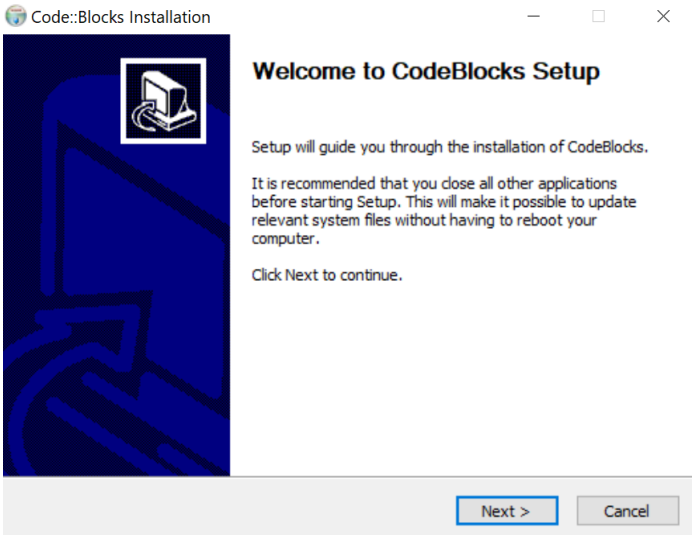
1. Instalarea mediului de programare

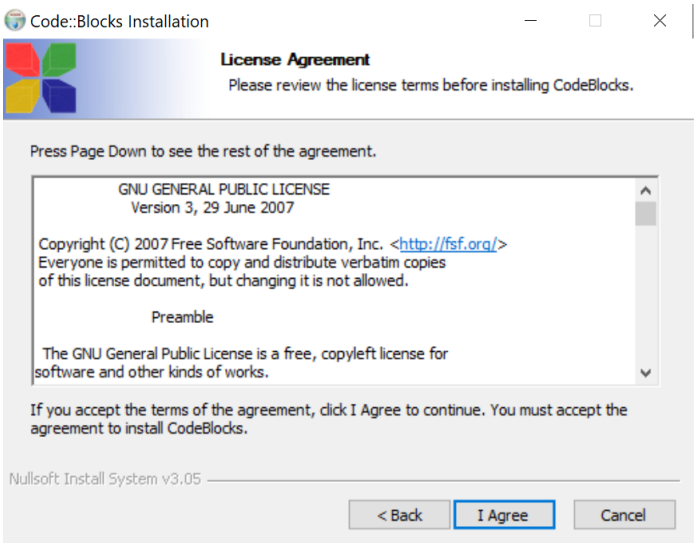
   - instalăm Code::Blocks împreună cu un compilator de c++

https://www.fosshub.com/Code-Blocks.html?dwl=codeblocks-20.03mingw-setup.exe

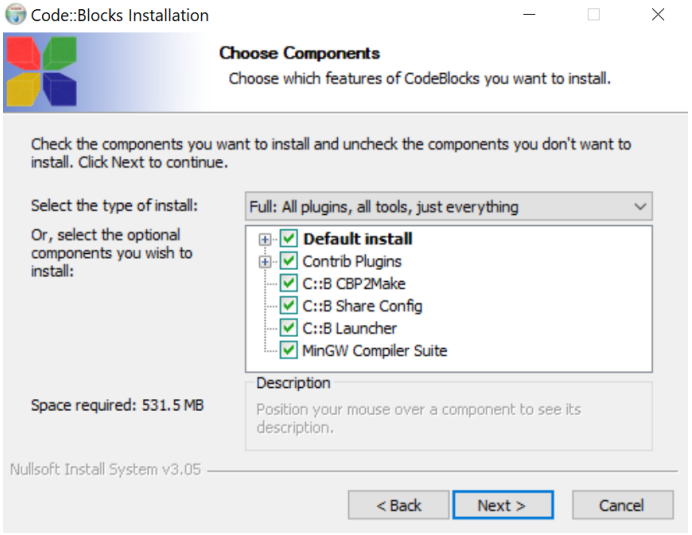
   - alegem versiunea care include compilatorul
   - ce face un compilator?

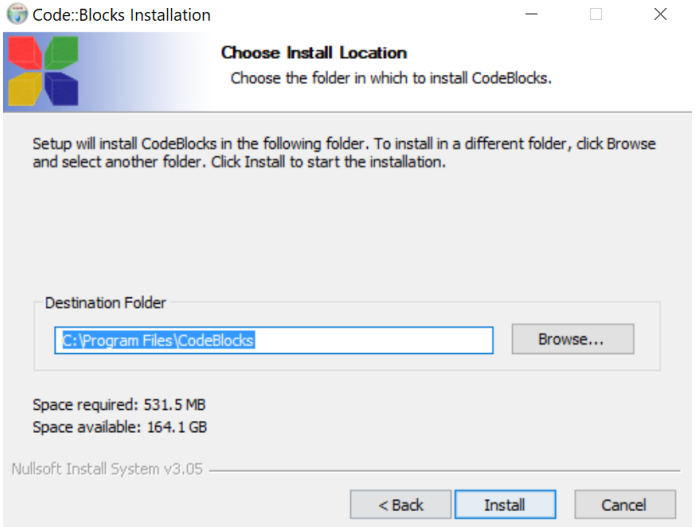




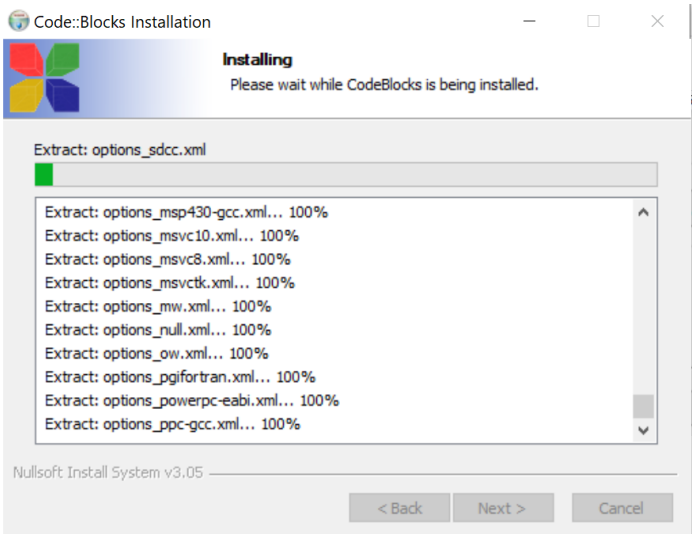


- lăsăm toate opțiunile bifate implicit



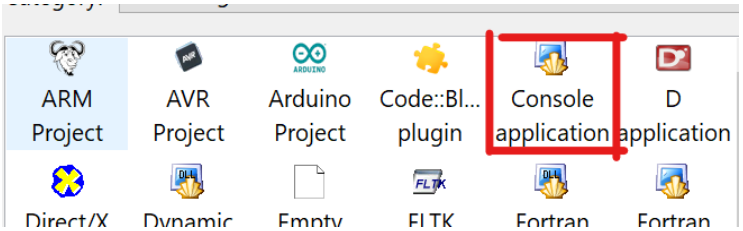


- după instalare, se poate deschide Code::Blocks automat

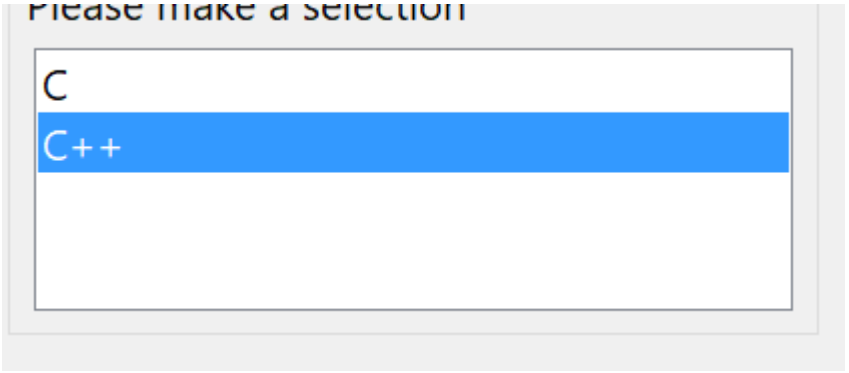


2. Crearea unui proiect nou de tip consolă
   - se creează pe desktop un folder nou cu numele vostru(acesta va fi folderul de lucru în care se găsesc toate proiectele)
   - crearea unui proiect nou de tip **consolă**

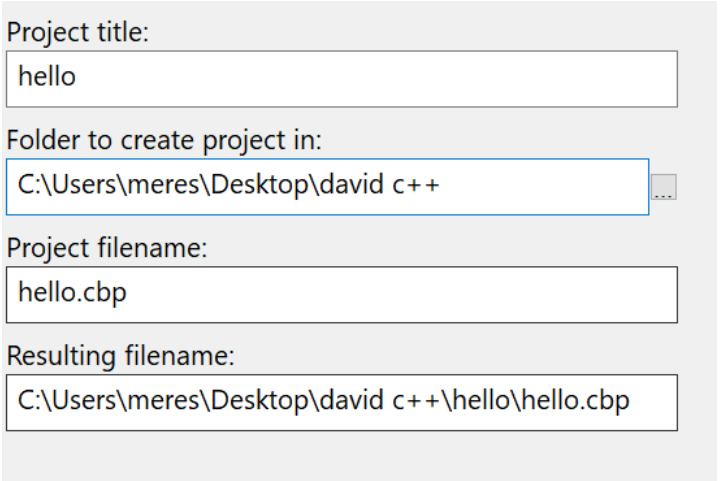
   - selectăm Console application



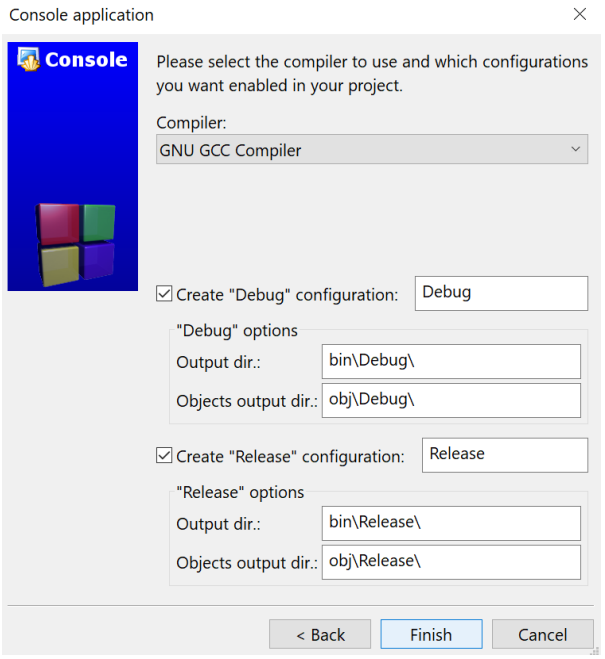
   - selectăm C++ și apăsăm Next



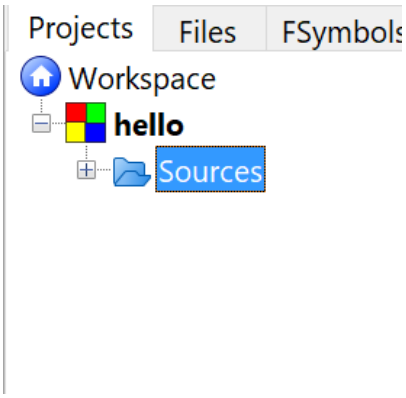
   - alegem un titlu de proiect în **Project title**
   - în căsuța **Folder to create project in** selectăm folderul cu numele elevului de desktop



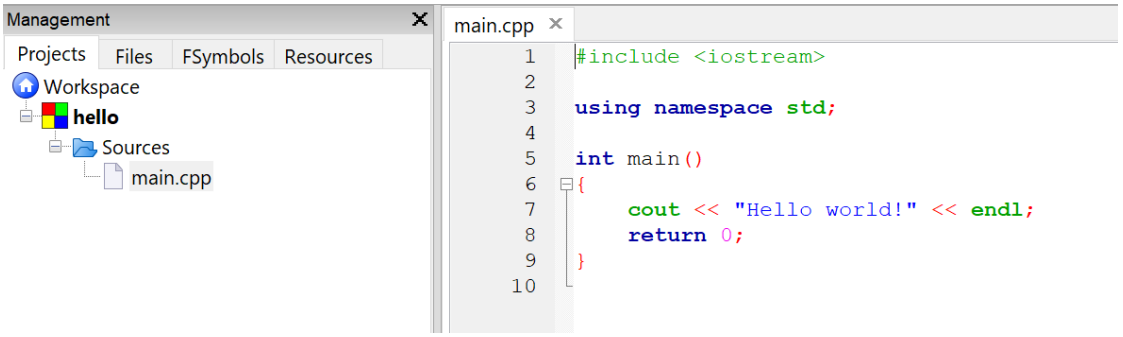
- apăsăm **Finish**



- în bara din stânga avem un explorer al proiectelor deschise



- în directorul **Sources** se găsește sursa de cod cu extensia **.cpp**



- pentru început nu trebuie să înțelegeți toate elementele din cod. Acestea vor fi explicate pas cu pas în lecțiile următoare.
- pentru a rula programul apăsăm tasta **F9** (sau **Fn + F9** pentru tastaturile care au **F9** scris mic sus).
- alternativ se poate rula apăsând pe rotița cu *play* din meniul de sus.

- în acest moment se deschide **consola** în care apare mesajul "Hello world!".

```
Hello world!

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

- consola se poate închide cu **X** după terminarea programului
- dacă modificăm linia de instrucțiune 7 și între ghilimele scriem un alt mesaj, acesta va apărea în consolă la următoarea rulare.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Eu sunt calculatorul si am fost programat sa-ti spun asta." << endl;
8      return 0;
9  }
10
```

```
Eu sunt calculatorul si am fost programat sa-ti spun asta.

Process returned 0 (0x0)   execution time : 0.162 s
Press any key to continue.
```

3. Ce este consola? Cum comunică un calculator?

   - orice mașină de calcul comunică cu utilizatorii printr-o interfață. Programele au diferite interfețe. Cele mai cunoscute interfețe sunt GUI (Graphical User Interface) care folosesc mouse-ul și tastatura ca principal mijloc de comunicare **input** și elementele grafice ca mijloc de comunicare **output**.

- input și output? ce înseamnă?

- aplicațiile pe care le vom crea în c++ în primă fază for folosi o **interfață de tip consolă**. Această interfață permite comunicarea cu calculatorul doar prin intermediul **textului** introdus (**input**) sau afișat (**output**) în consolă.
- se rulează următorul program ca exemplu, fără a explica codul din spate (doar funcționarea este esențială).

```
#include <iostream>

using namespace std;

int main()
{
    cout<<"introdu orice numar>>";
    int nr;
    cin>>nr;
    if ( nr % 2 == 0){
        cout<<"numarul "<<nr<<" este par!"<<endl;
    }else{
        cout<<"numarul "<<nr<<" este impar!"<<endl;
    }
    return 0;
}
```

4. Aplicații
   a. Salut, Andrei!
      - se importă biblioteca <string>.
      - se declară o variabilă string nume;
      - se citește variabila de la tastatură, citirea este însoțită mereu de un mesaj sugestiv.
      - se afișează conținutul variabilei împreună cu mesajul de salut.
      - din nou, nu trebuie să înțelegeți toate elementele din cod. Este suficient să urmăriți fluxul de input și output al programului.

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

int main()
{
    string nume;
    cout<<"cum te cheama?"<<endl<<">>";
```

```
    cin>>nume;
    cout<<"Salut, "<<nume<<"!"<<endl;
    return 0;
}
```

b. ASCII ART

- veți afișa în consolă tot felul de desene construite din caractere ASCII (litere, cifre și simboluri).
- se poate accesa site-ul ASCII Art Archive de unde se pot alege modele.
- fiecare **linie** trebuie copiată (CTRL+C) din construcțiile ASCII ART și introdusă (CTRL+V) între ghilimelele de la **cout<<"----"<<endl;** ca în exemplul de mai jos.

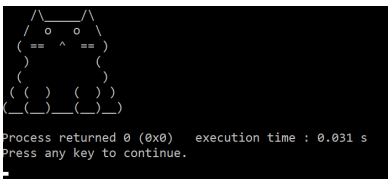
```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    cout<<"     /\\_____/\\"<<endl;
    cout<<"    /  o   o  \\"<<endl;
    cout<<"   ( ==  ^  == )"<<endl;
    cout<<"    )         ("<<endl;
    cout<<"   (           )"<<endl;
    cout<<"  ( (  )   (  ) )"<<endl;
    cout<<"(__(__)___(__)__)"<<endl;
    return 0;
}
```

- **mare atenție.** Unele caractere cum ar fi

  ` " \ ; '

  aparțin sintaxei c++. Astfel, compilatorul le interpretează ca elemente de cod. Pentru a evita astfel de erori, înaintea oricărui astfel de caracter se poate pune un \ (escape character) pentru a anula efectul sintactic al respectivului caracter.
  - compilatorul vă sugerează cu roșu linia la care s-a întâlnit o eroare.
- de exemplu, se observă cum în c++ între ghilimele se scrie un text (string). Dar dacă imaginea noastră ASCII ART conține ghilimele, asta va genera o eroare de sintaxă în cod. Dacă punem un \ înainte de " efectul lor de delimitator de string va dispărea.
- se observă cum în exemplu, fiecare \ este precedat de un alt \ care îi anulează efectul.



- în consolă, acel caracter \ pus în plus nu mai apare.