ĐỒ HỌA MÁY TÍNH

Hướng dẫn cài đặt và sử dụng CodeBlocks

Mục lục

1 Cài đặt

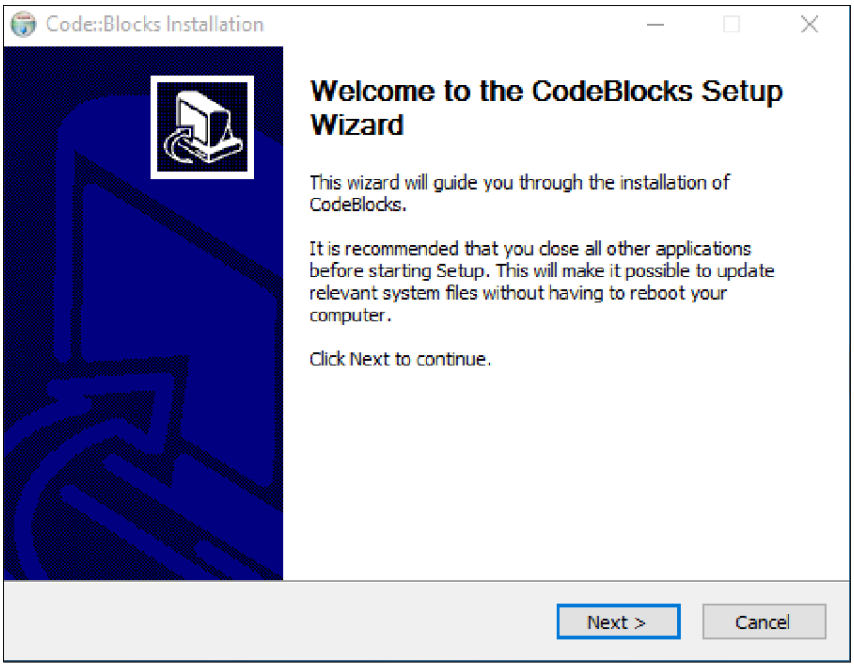
1.1 Download

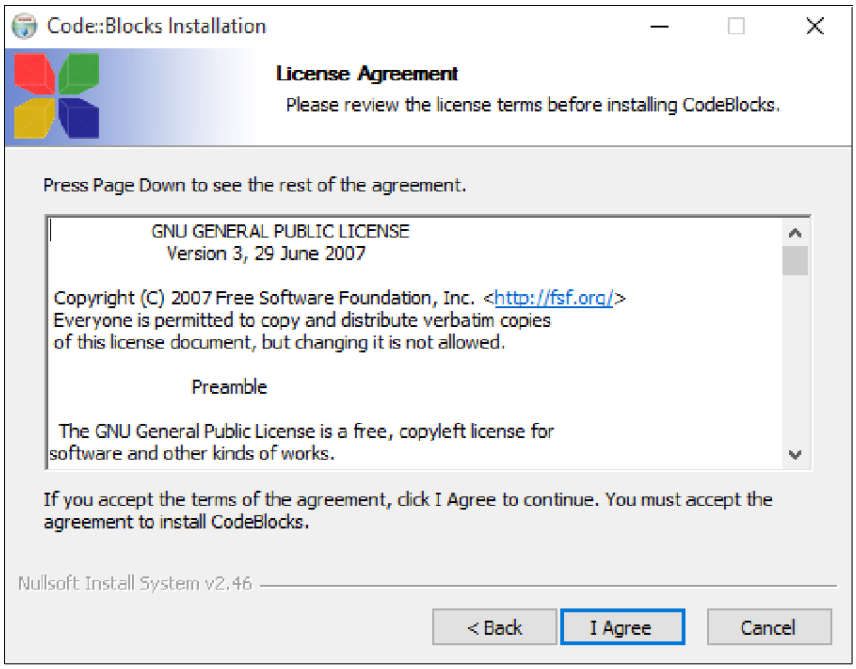
- Down load chương trình cài đặt CodeBlocks 13.12 tại đường link sau: https://www.mediafire.com/file/9o8d0bfwdnsinfe/codeblocks-setup.exe/file.
- Down load thư viện đồ họa tại đường link sau:

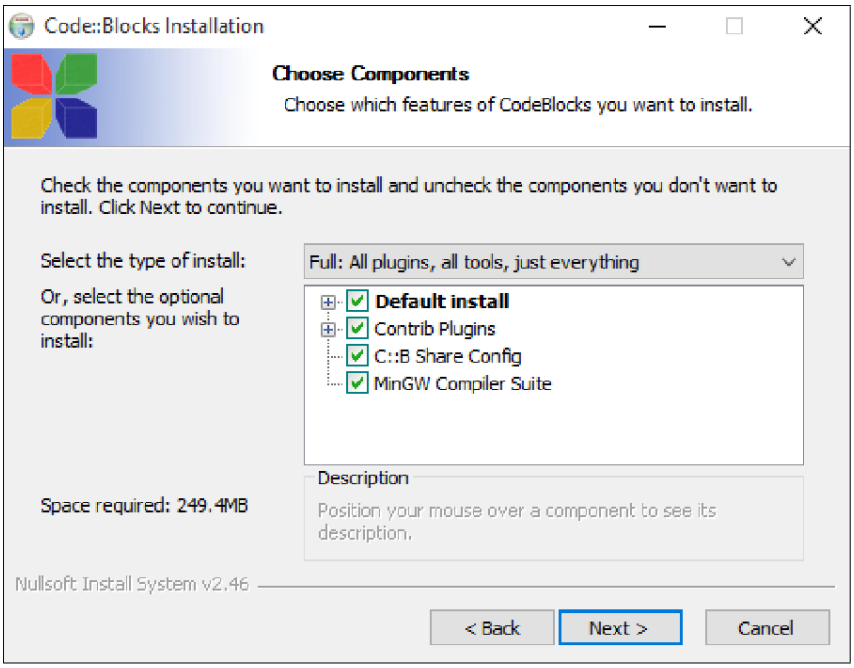
  https://www.mediafire.com/file/1ia4iylxdbuxv0l/OpenGLforCodeBlocks.rar/file

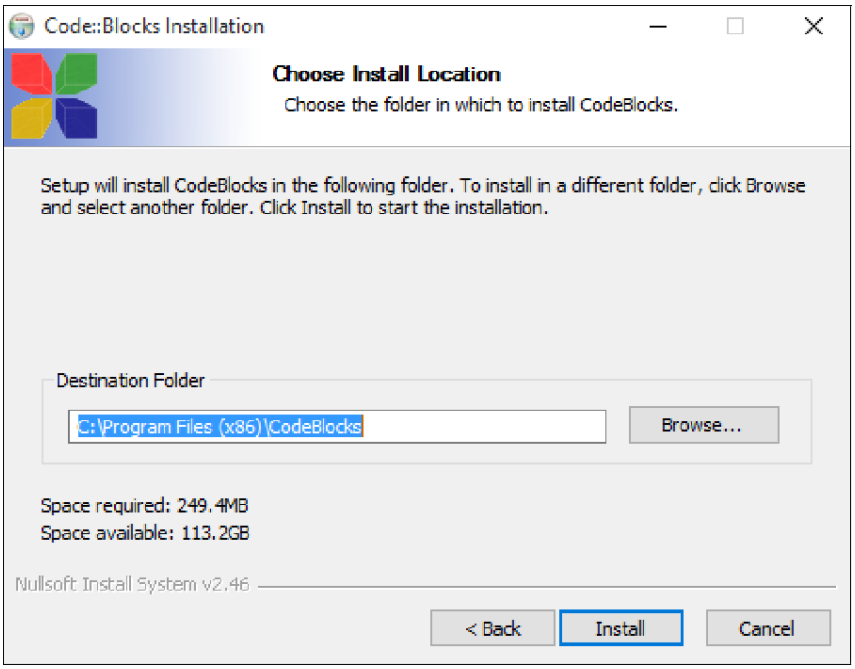
1.2 Cài đặt

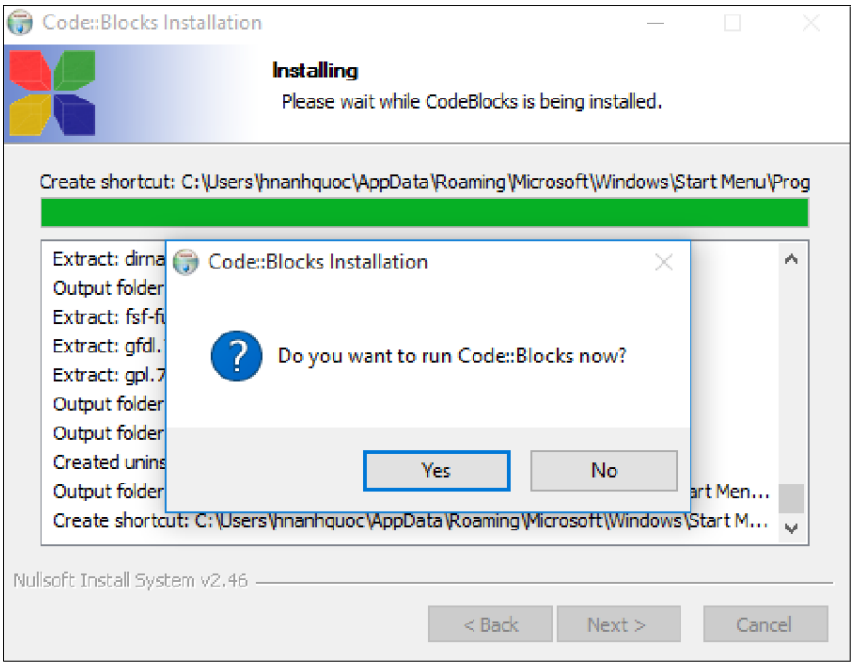
1. Chạy file vừa tải về (codeblocks-13.12mingw-setup.exe) với các cài đặt mặc định.



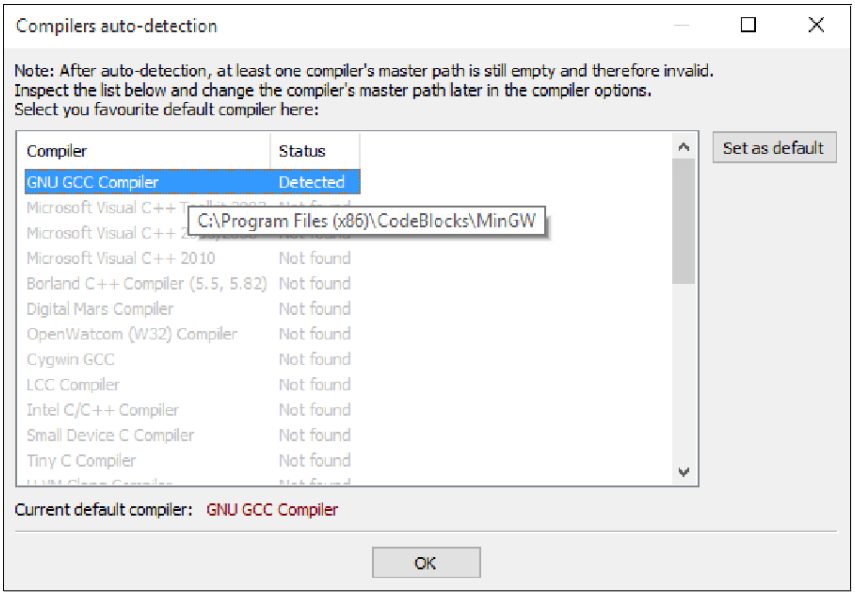



Code::Blocks Installation
Choose Components
Choose which features of CodeBlocks you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select the type of install:
Full: All plugins, all tools, just everything
Or, select the optional components you wish to install:
Default install
Contrib Plugins
C::B Share Config
MinGW Compiler Suite
Description
Position your mouse over a component to see its description.
Space required: 249.4MB
Nullsoft Install System v2.46
< Back
Next >
Cancel

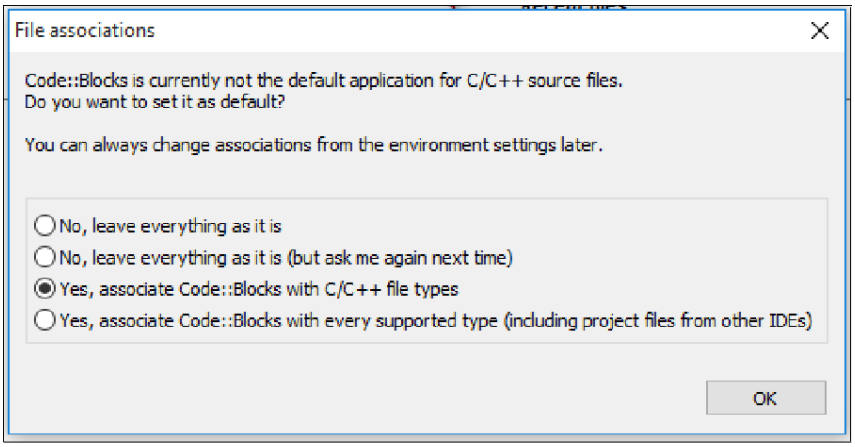

Code::Blocks Installation
Choose Install Location
Choose the folder in which to install CodeBlocks.
Setup will install CodeBlocks in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.
Destination Folder
C:\Program Files (x86)\CodeBlocks
Browse...
Space required: 249.4MB
Space available: 113.2GB
Nullsoft Install System v2.46
< Back
Install
Cancel



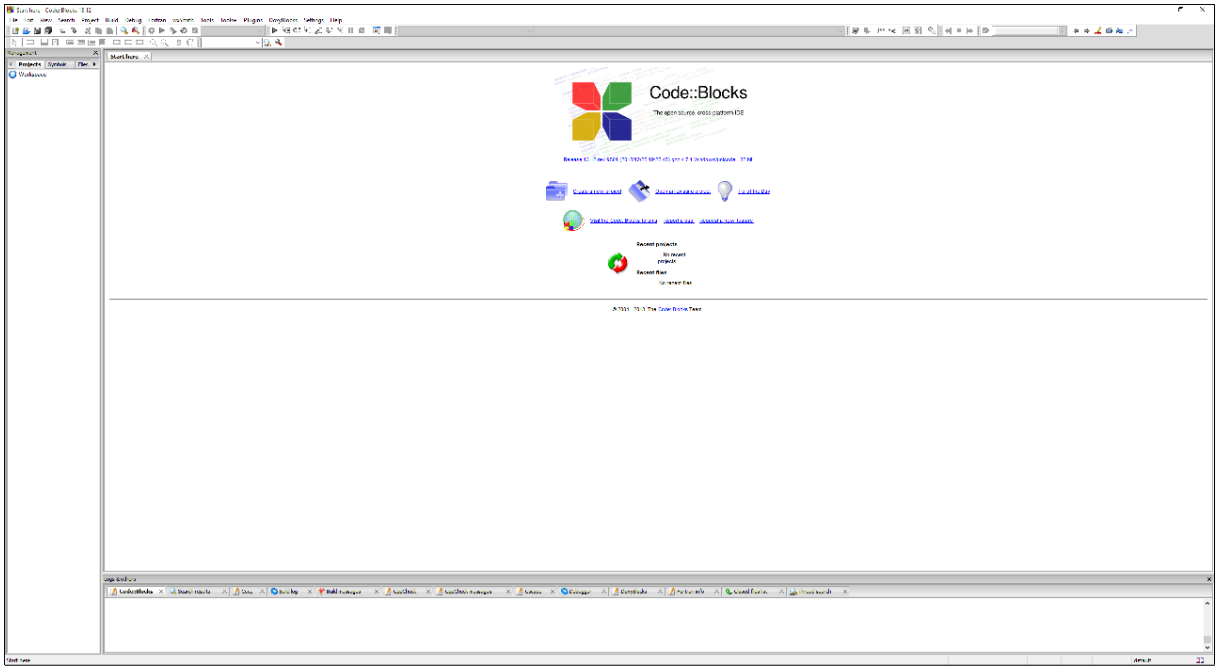
Chọn “Yes” để mở chương trình Code::Blocks. Sẽ xuất hiện thông báo để chọn trình biên dịch và chúng ta để mặc định



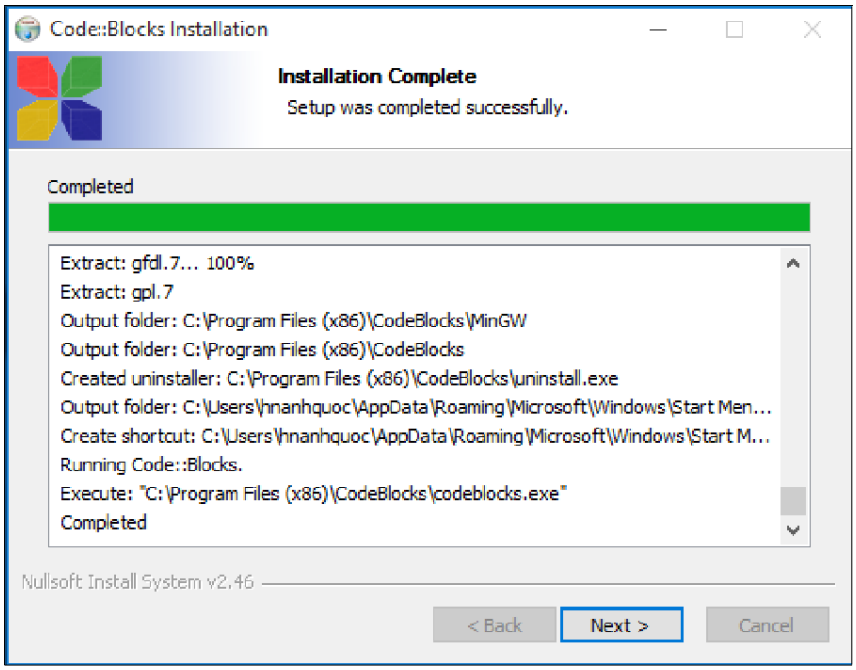
Xuất hiện thông báo hỏi về việc sử dụng Code::Blocks cho các file định dạng .c và .cpp. Nên để mặc định và chọn “OK”.

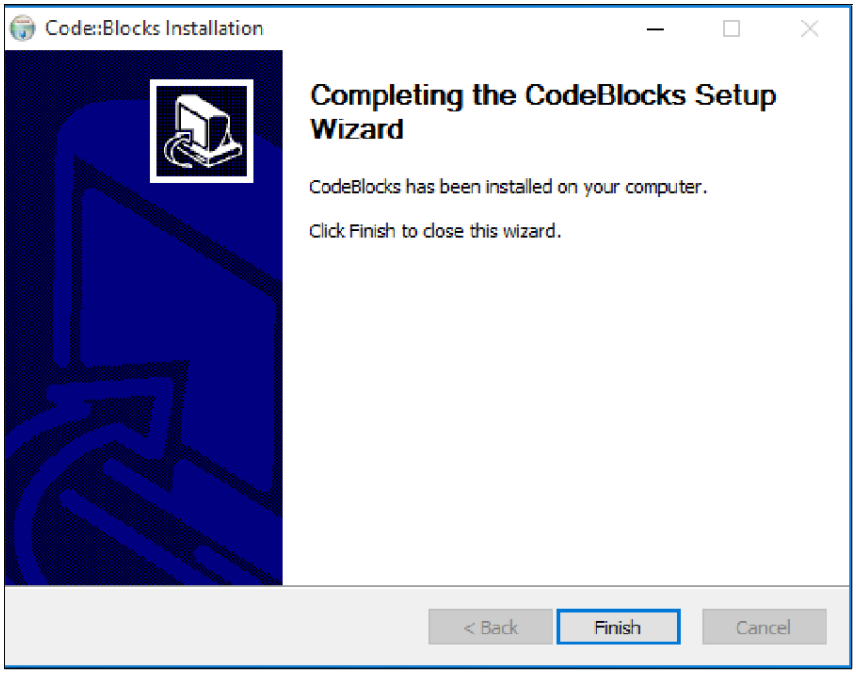


Chương trình có giao diện như sau:



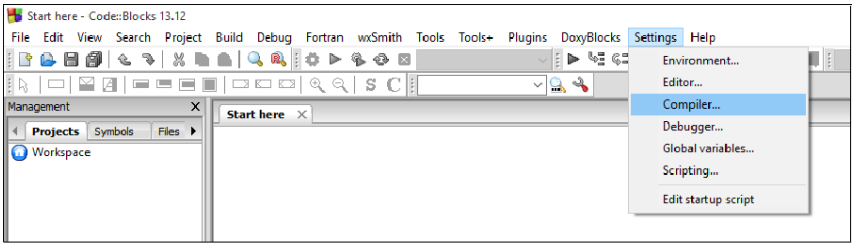
Sau khi chương trình đã mở, chúng ta quay lại trình cài đặt để hoàn tất quá trình cài đặt

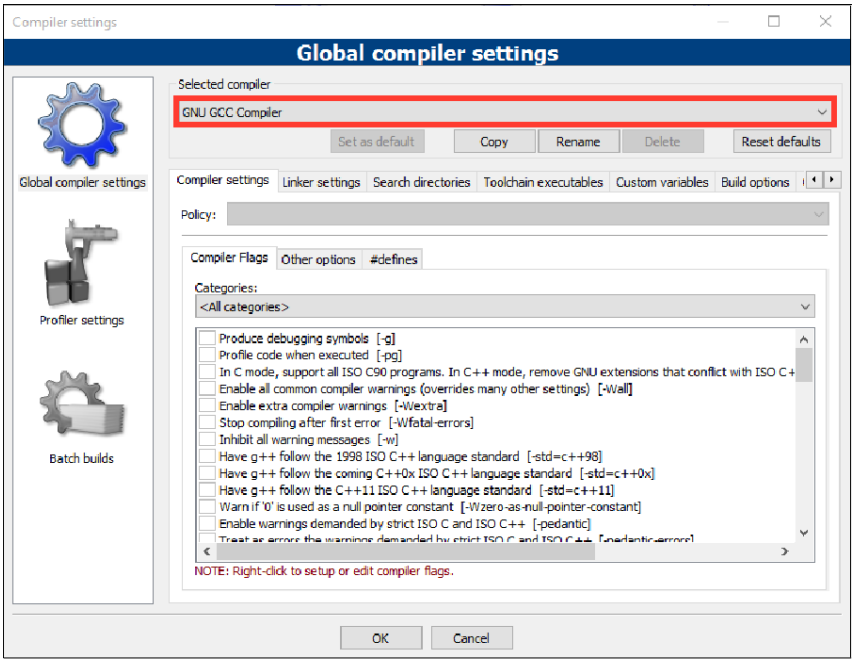




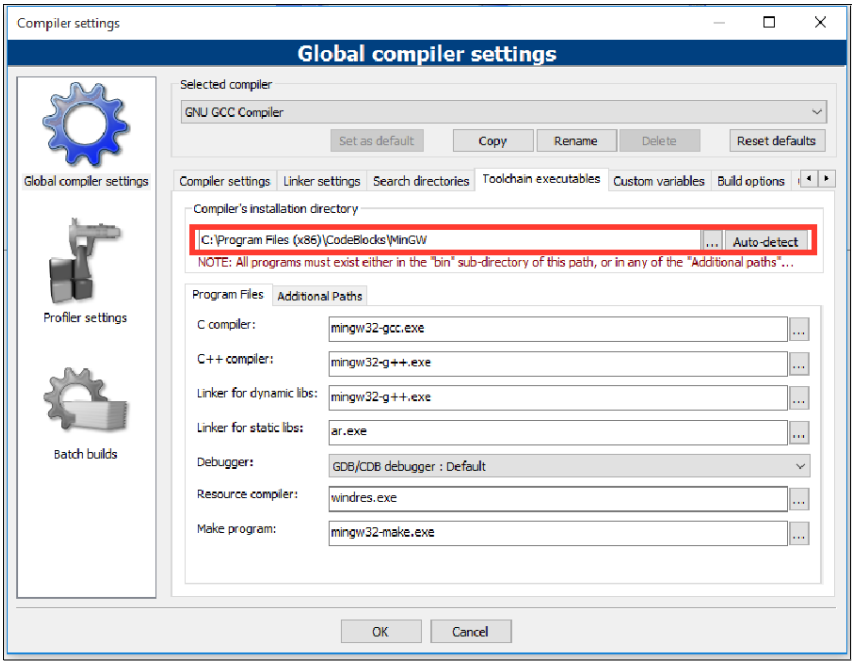
2. Kiểm tra các đường dẫn

(a) Kiểm tra đường dẫn của Compiler

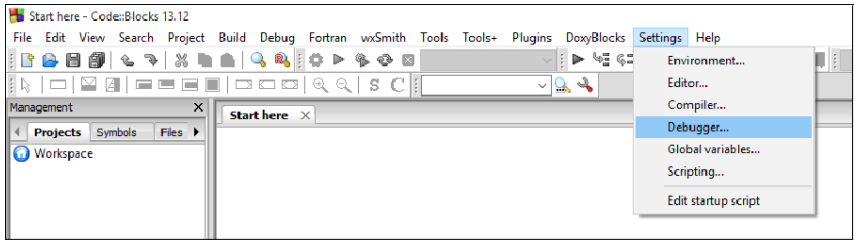


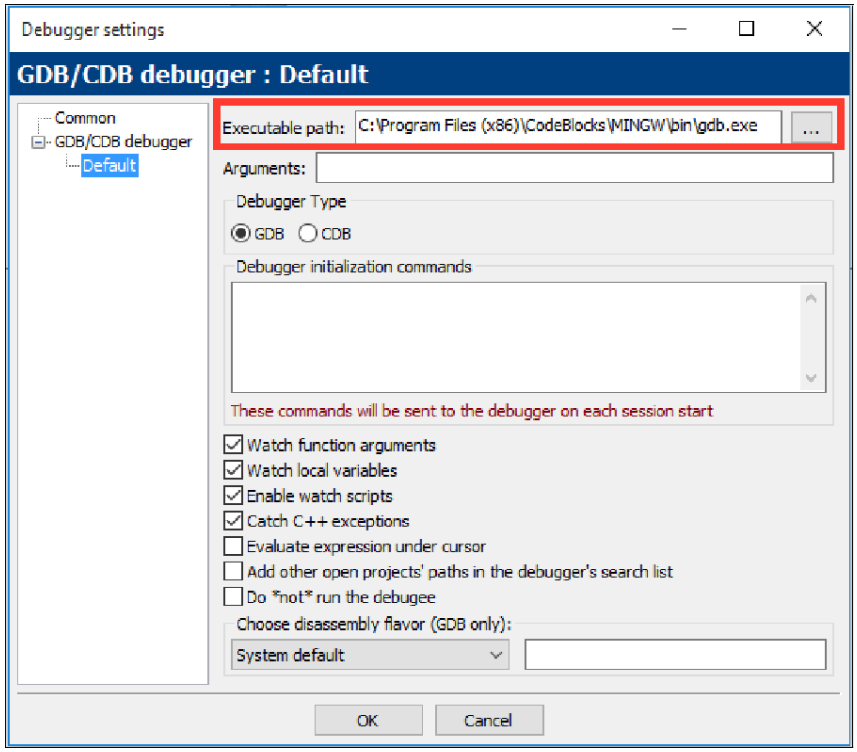


Chọn thẻ "Toolchain Executables".



(b) Kiểm tra đường dẫn của Debugger

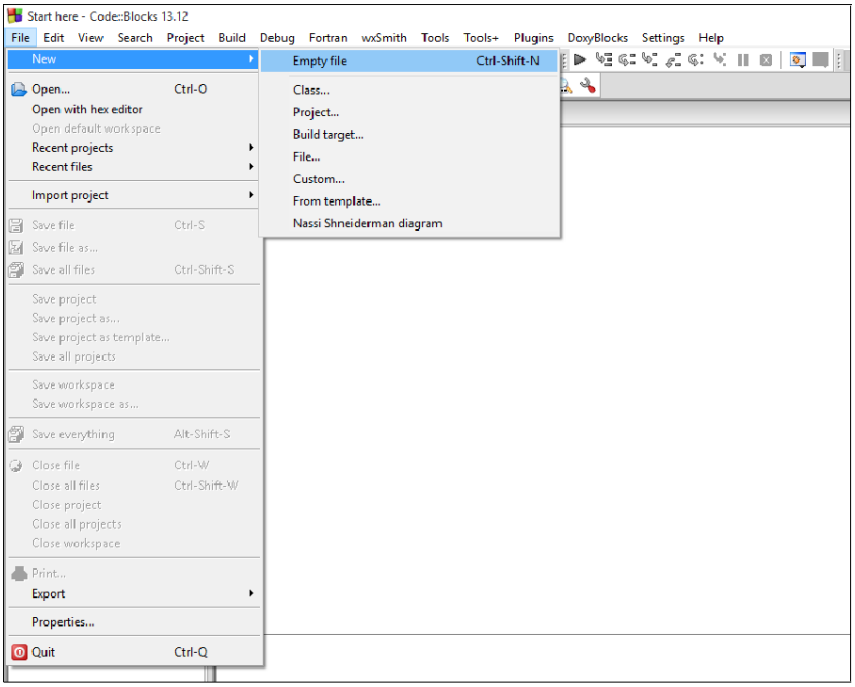




2 Viết chương trình C/C++ với CodeBlocks

2.1 Viết chương trình không cần tạo project

1. Tạo file

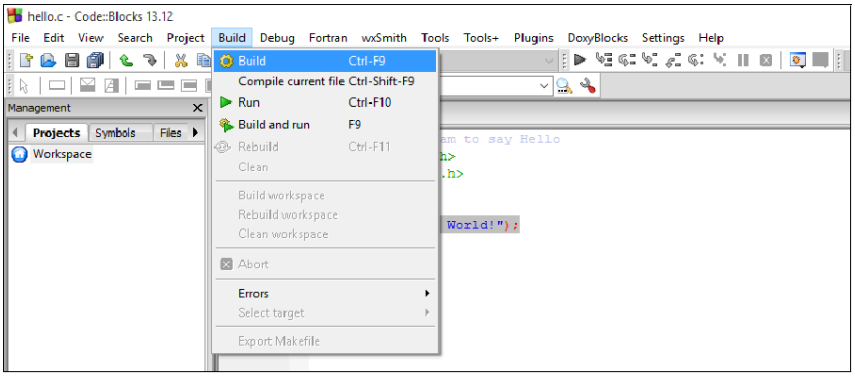


2. Nhập đoạn code sau:

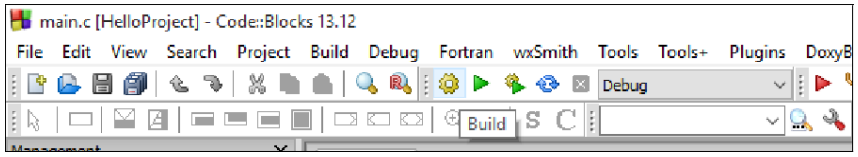
```
1  // First C program to say Hello
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  void main() {
6      printf("Hello World!");
7  }
8
```

3. Lưu file với tên "hello.c" vào một thư mục nào đó (ví dụ như C:\MyProjects)

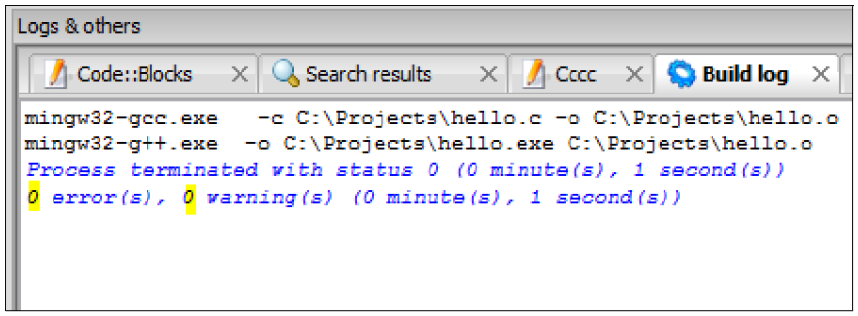
4. Build chương trình



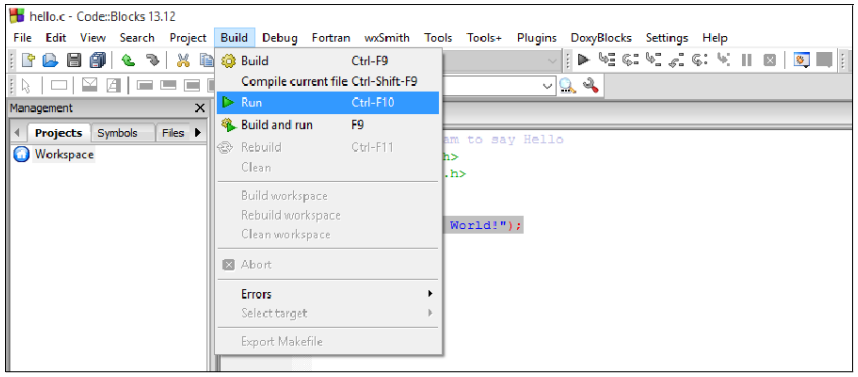
Hoặc sử dụng phím tắt trên thanh công cụ:



Chương trình Build thành công sẽ có thông báo như sau (bên dưới màn hình):

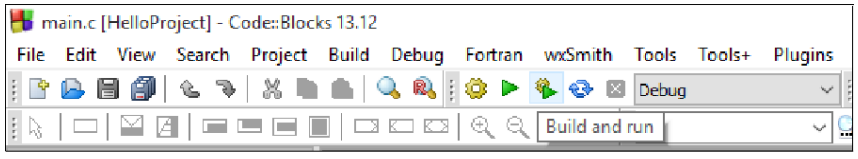


5. Chạy chương trình

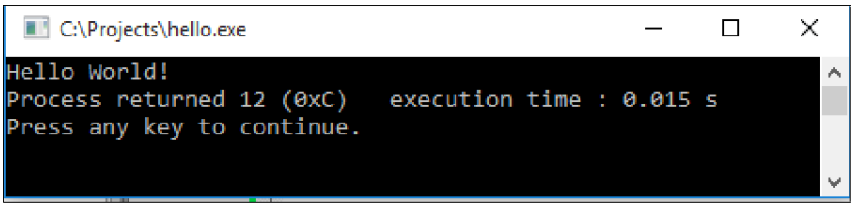


Tương tự, ta cũng có thể sử dụng phím tắt:

Hoặc



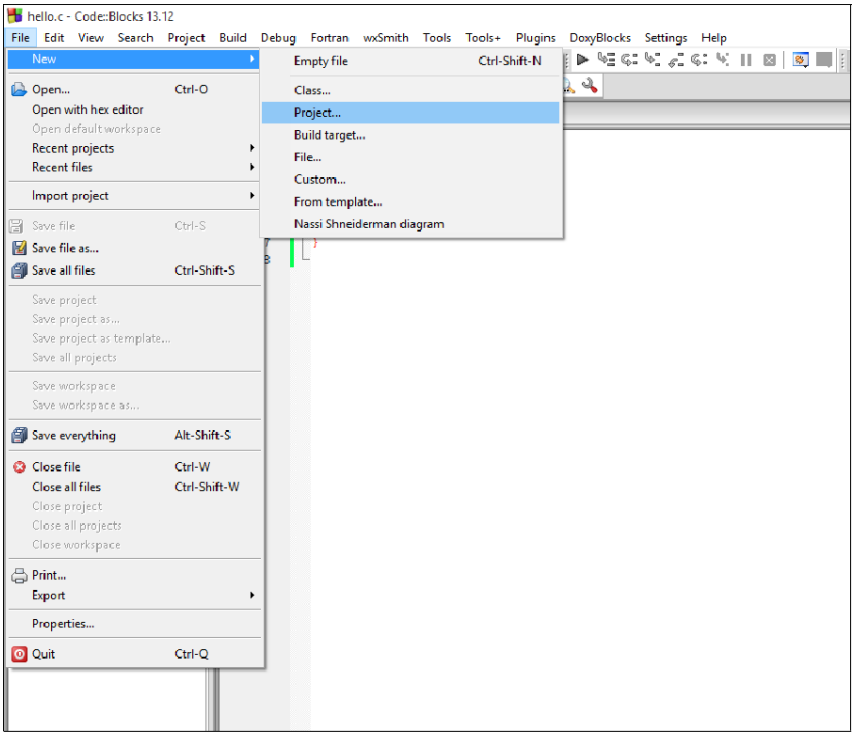
6. Kết quả



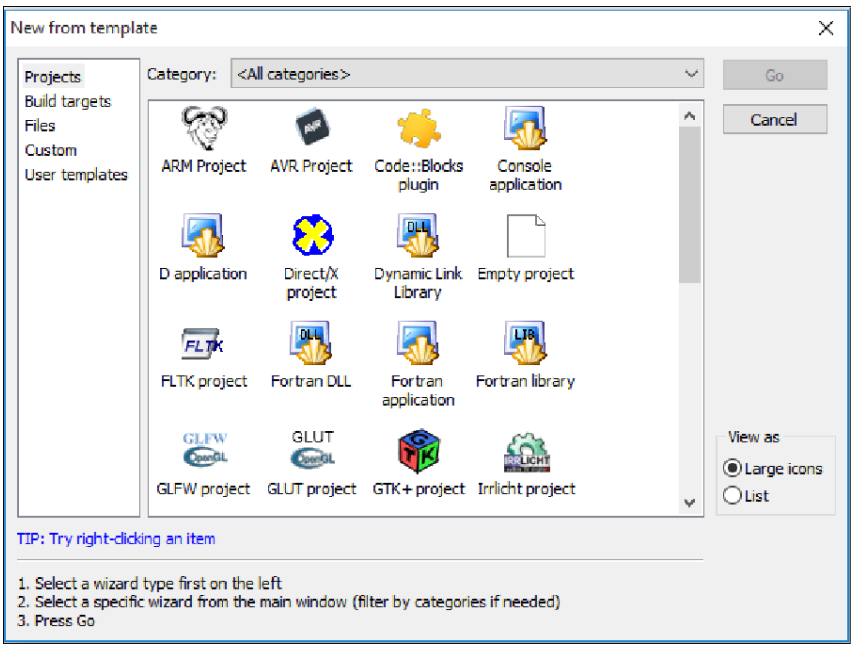
- **Ưu điểm:** Nhanh chóng và tiện lợi.
- **Nhược điểm:** Không thể debug khi có lỗi xảy ra. Để có thể debug, chúng ta phải sử dụng cách thứ 2: viết chương trình trong một project.

2.2 Viết chương trình trong một project

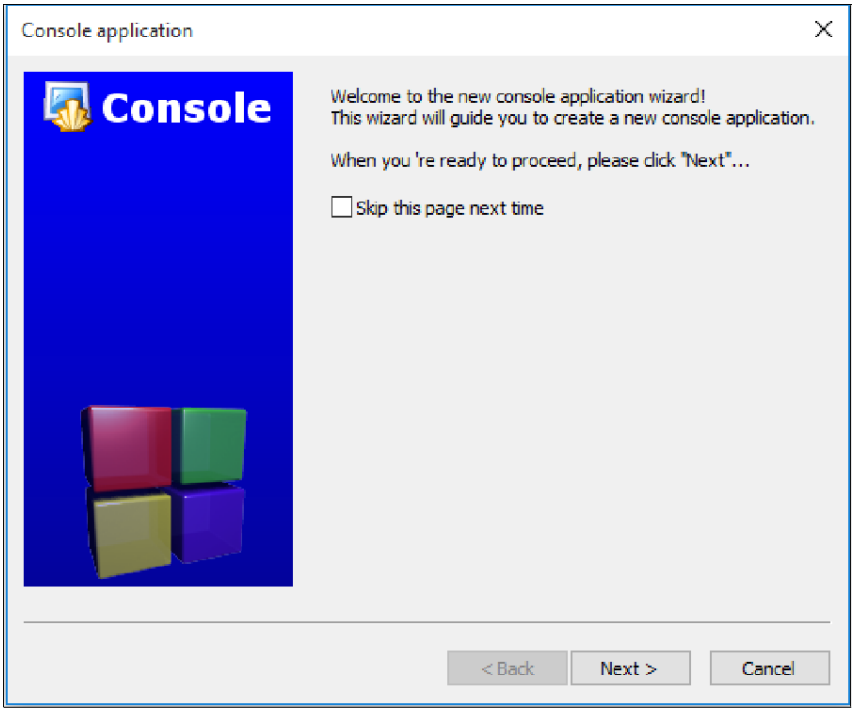
1. Tạo project mới



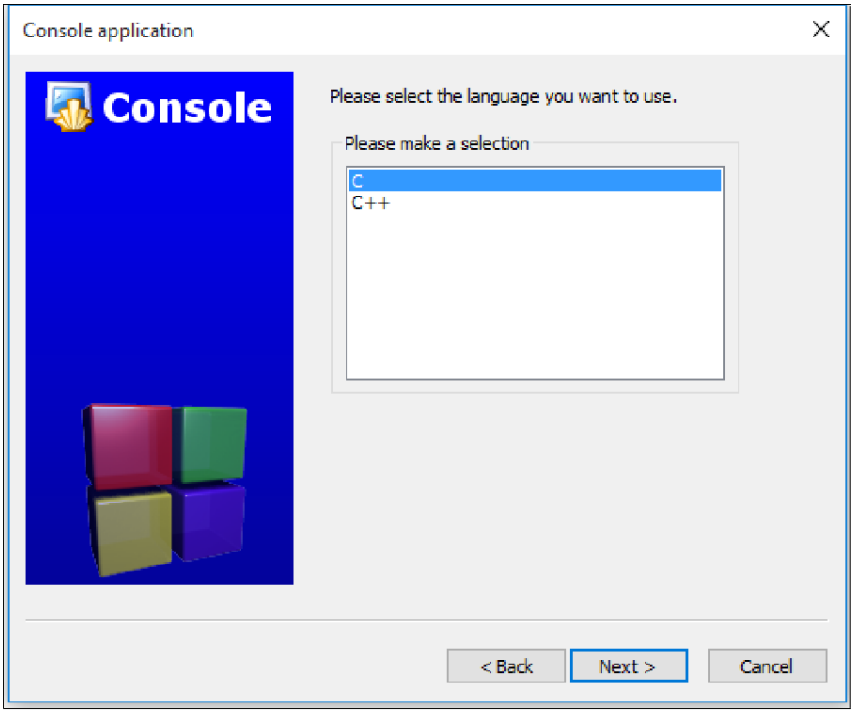
Chọn "Console Application", "Go"

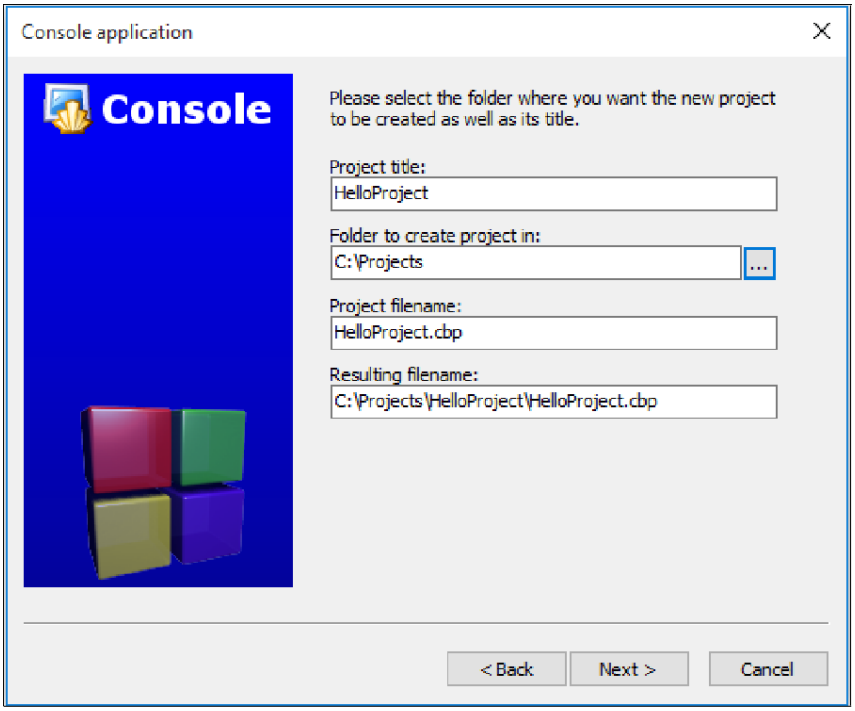


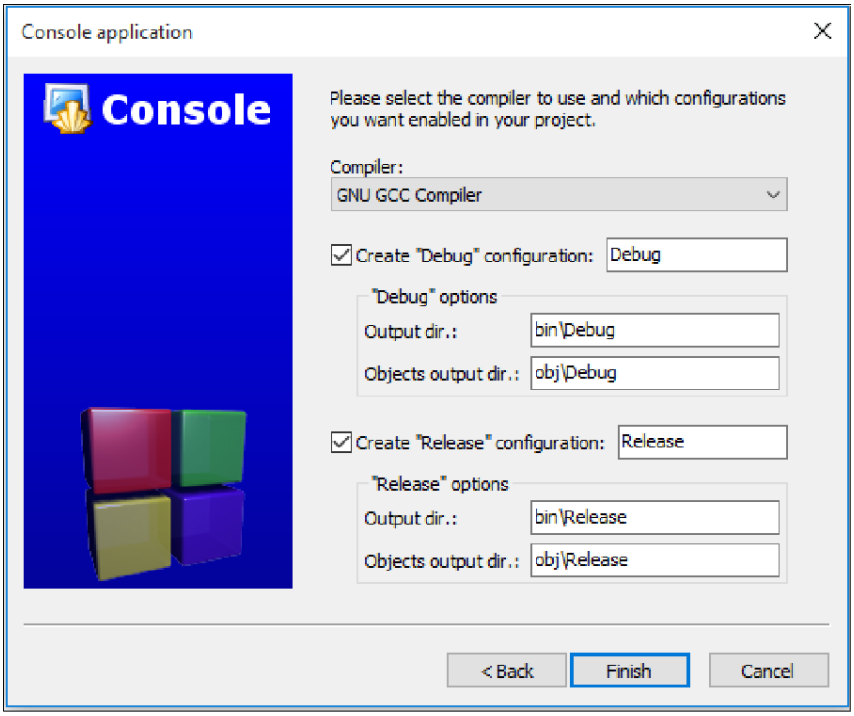
Nhấn "Next"



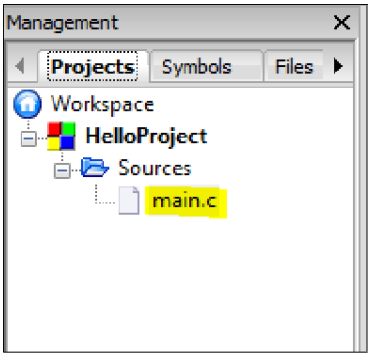
Chọn "C" và nhấn "Next"


Console application
Console
Please select the language you want to use.
Please make a selection
C
C++
< Back
Next >
Cancel


Console application
Console
Please select the folder where you want the new project to be created as well as its title.
Project title:
HelloProject
Folder to create project in:
C:\Projects
...
Project filename:
HelloProject.cbp
Resulting filename:
C:\Projects\HelloProject\HelloProject.cbp
< Back
Next >
Cancel

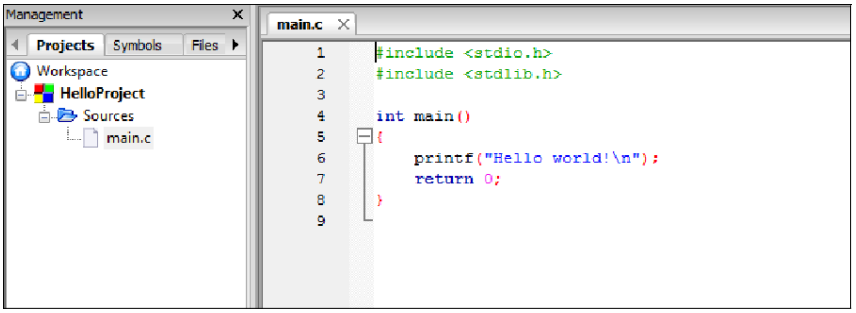


Sau khi hoàn tất, trên thanh "Management" nằm ở bên trái sẽ xuất hiện project vừa tạo:



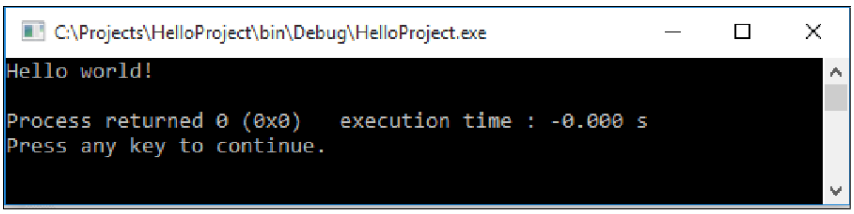
2. Source files

Nhấn đúp vào "main.c". Đây là code do chương trình sinh ra:

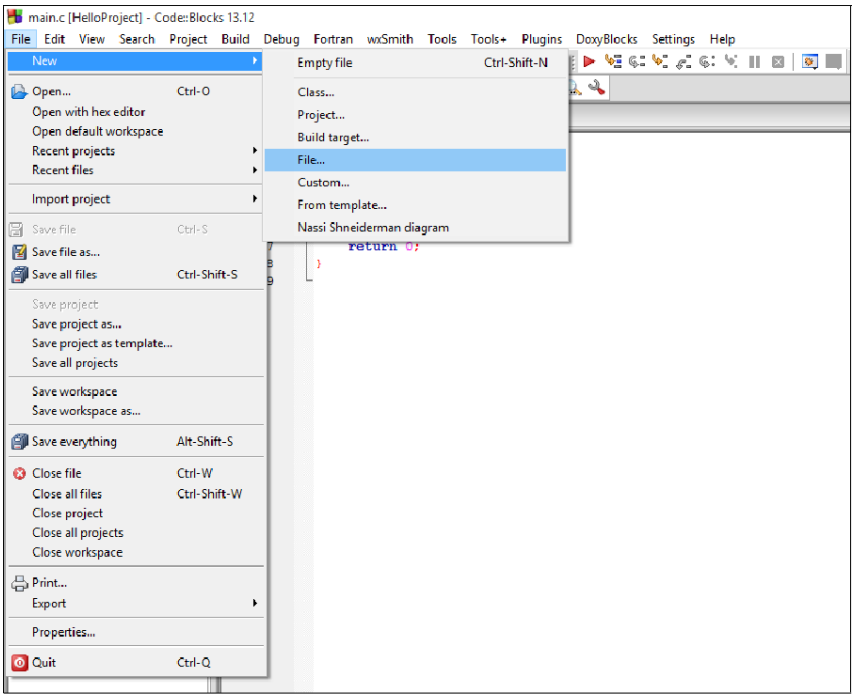


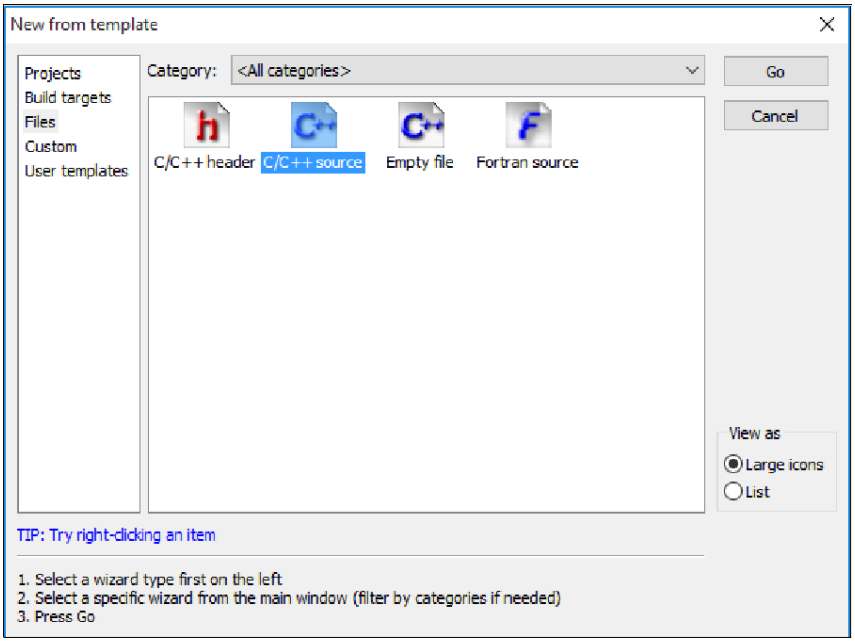
3. Build và chạy chương trình

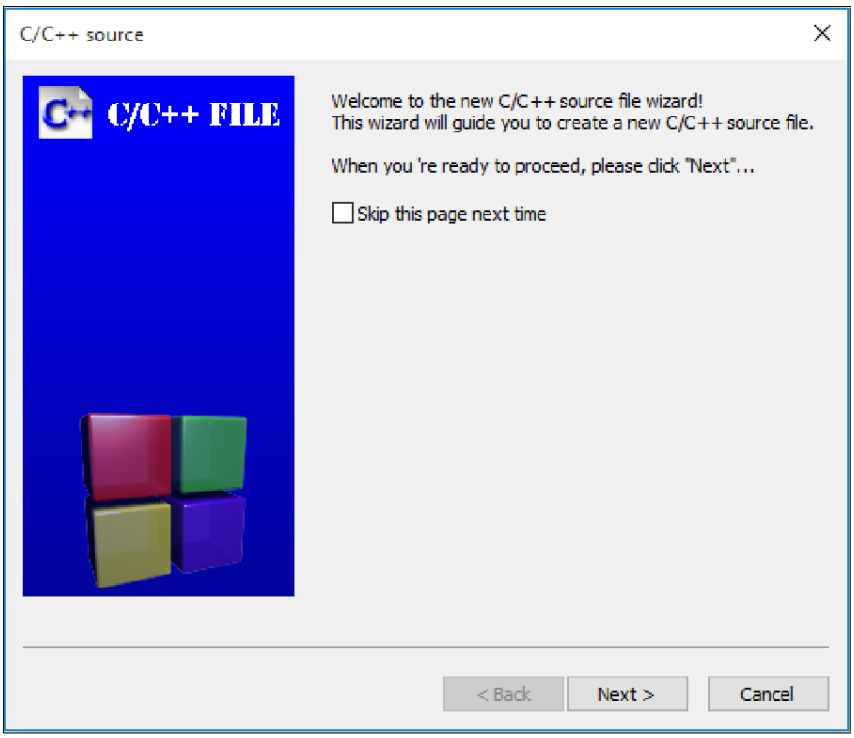
Tương tự như thao tác đối với chương trình không cần tạo project.

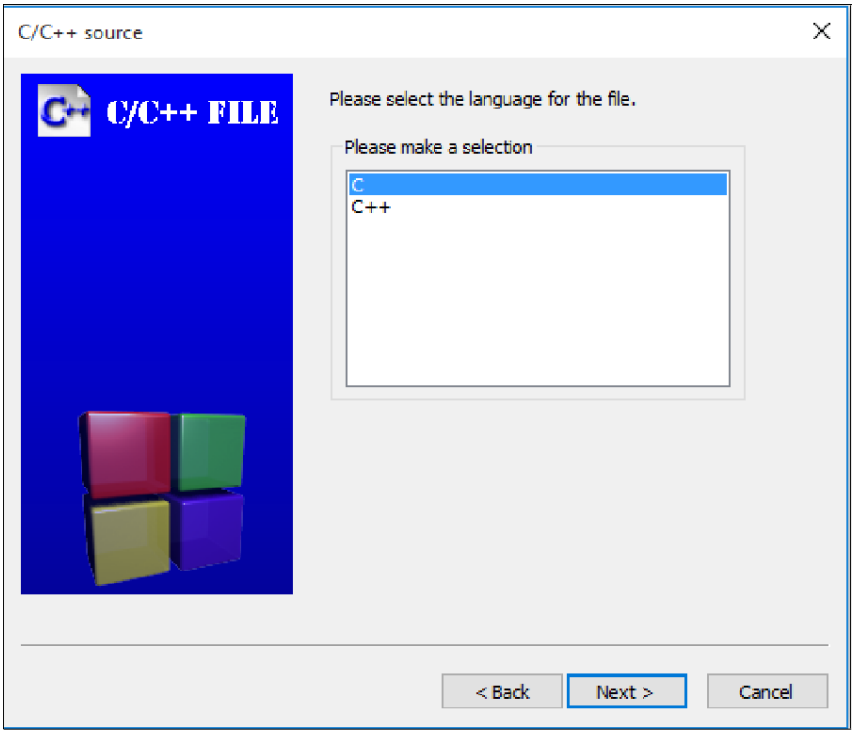


4. Thêm các source/header files

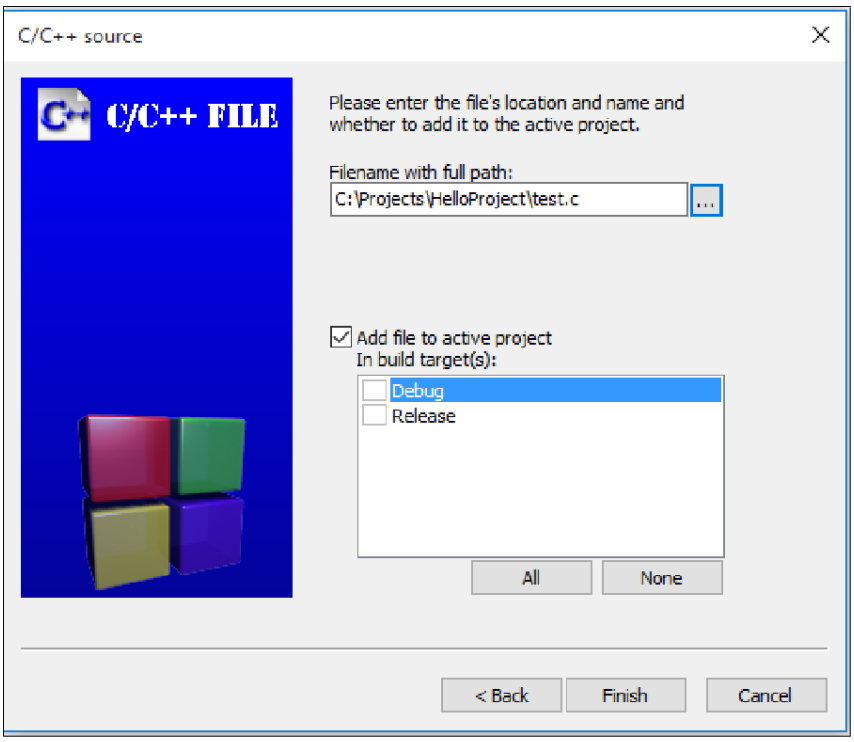




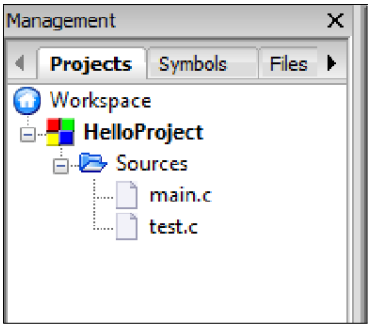




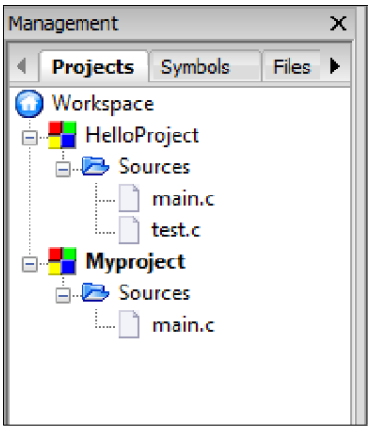
Chọn đường dẫn tới thư mục của project hiện tại và đặt tên:

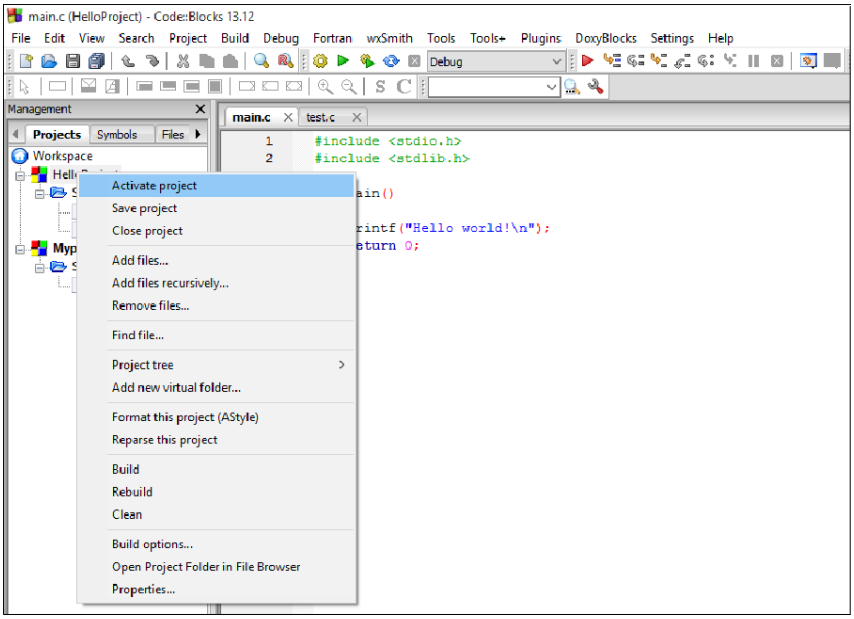


Kết quả:



Đối với trường hợp có nhiều project thì phải thực hiện “Set Active” cho project mà ta muốn Build và Run.




main.c (HelloProject) - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects Symbols Files
Workspace
main.c
test.c
#include <stdio.h>
#include <stdlib.h>
Activate project
Save project
Close project
Add files...
Add files recursively...
Remove files...
Find file...
Project tree
Add new virtual folder...
Format this project (AStyle)
Reparse this project
Build
Rebuild
Clean
Build options...
Open Project Folder in File Browser
Properties...

3 Debug chương trình C++ với CodeBlocks

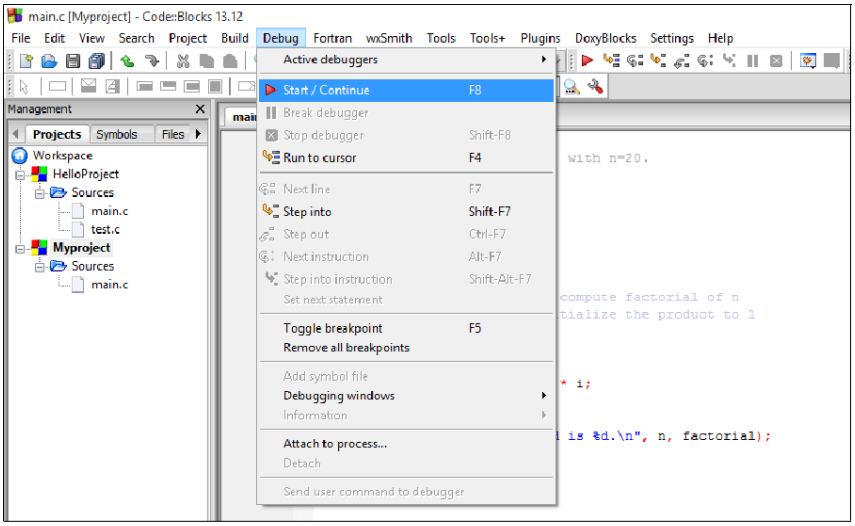
1. Viết chương trình C
   Nhập đoạn code sau vào file main.c:

```c
/*
 * Compute the factorial of n, with n=20.
 *    n! = 1*2*3*...*n
 */
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n = 20;          // To compute factorial of n
    int factorial = 1;   // Initialize the product to 1

    int i = 1;
    while (i <= n){
        factorial = factorial * i;
        i++;
    }
    printf("The Factorial of %d is %d.\n", n, factorial);
    return 0;
}
```

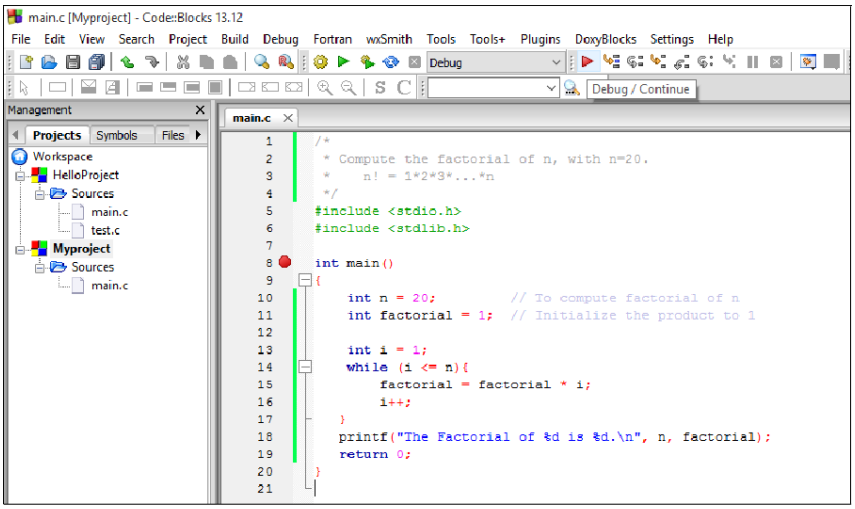
2. Đặt các breakpoints
   Breakpoint là các điểm mà chúng ta muốn chương trình dừng lại tại đó. Để đặt breakpoint, ta nhấn vào khoảng trắng sau số dòng. Breakpoint được hiển thị là chấm màu đỏ, một chương trình có thể có một hoặc nhiều breakpoint.

```c
int main()
{
    int n = 20;          // To compute factorial of n
    int factorial = 1;   // Initialize the product to 1
```

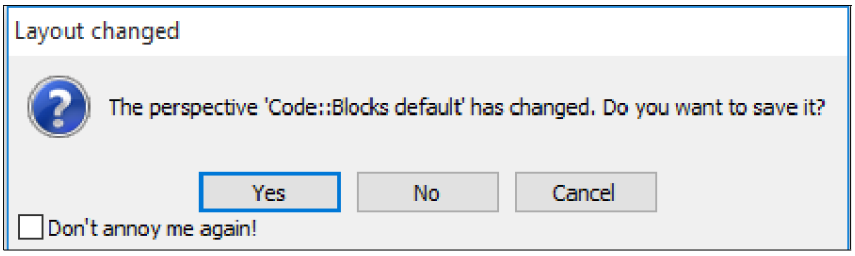
3. Chạy debug



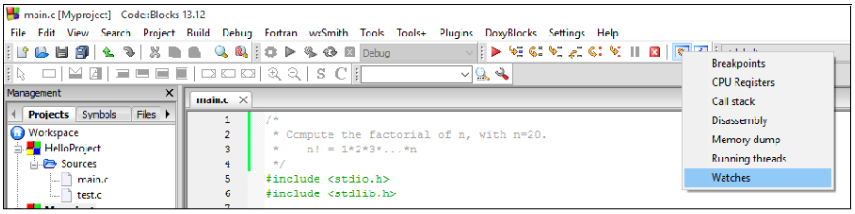
Hoặc

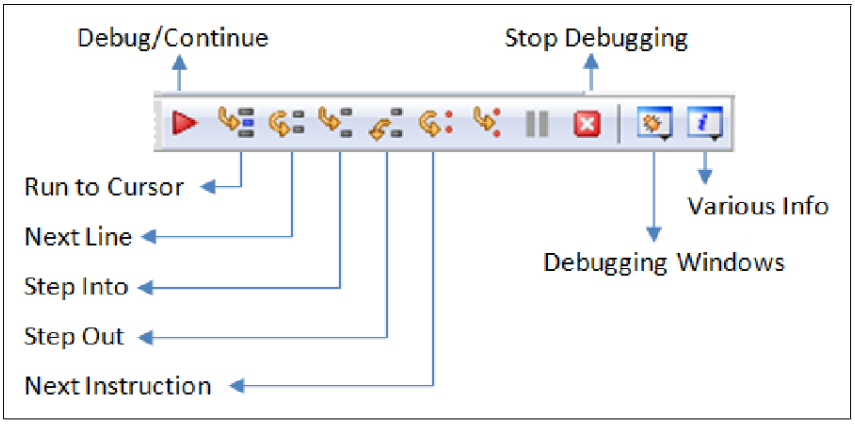


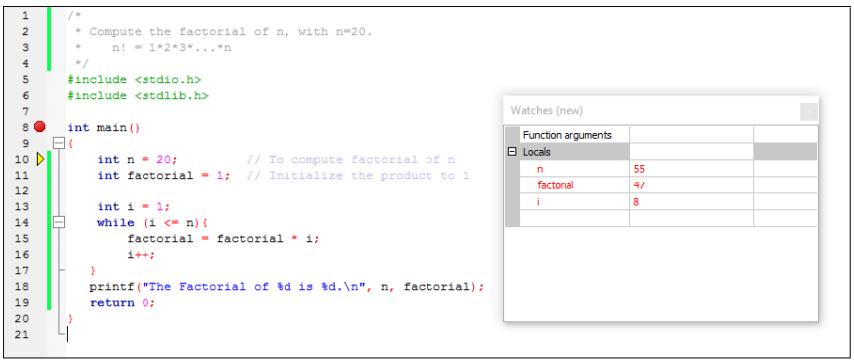
Một hộp thoại xuất hiện, hỏi rằng có muốn chuyển sang giao diện debug hay không. Chọn “Yes”.

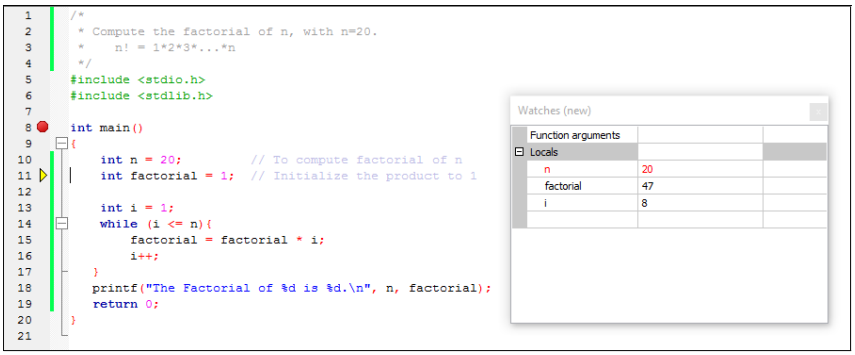


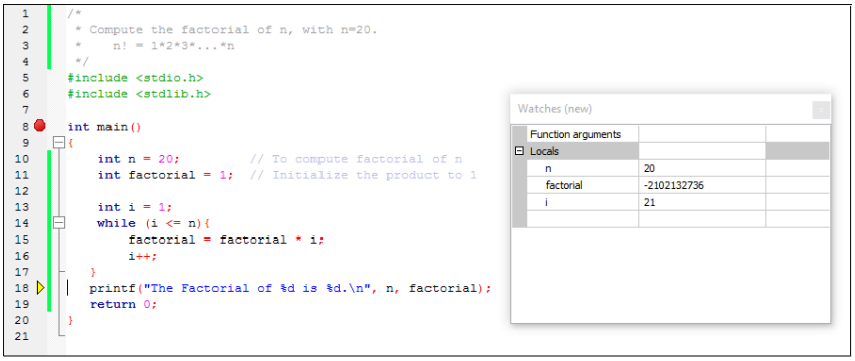
4. Bật các cửa sổ theo dõi











Quan sát giá trị các biến có thể giúp ta phát hiện được các lỗi của chương trình.

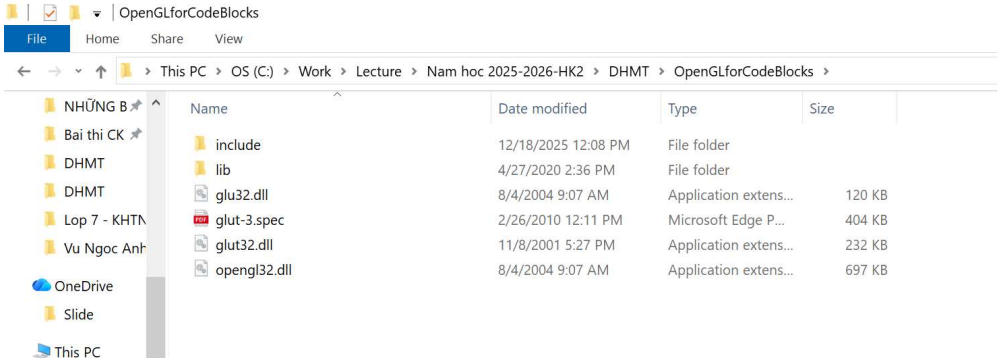
4 Lập trình đồ họa với CodeBlocks

1. Cài đặt thư viện đồ họa

Bước 1: Download thư viện đồ họa theo đường dẫn :

- https://www.mediafire.com/file/1ia4iylxdbuxv0l/OpenGLforCodeBlocks.rar/file

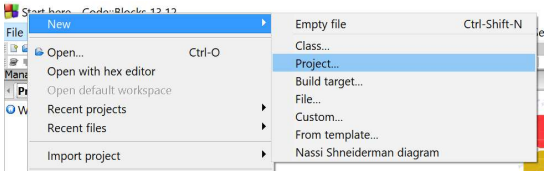
Bước 2: Giải nén file vừa download:

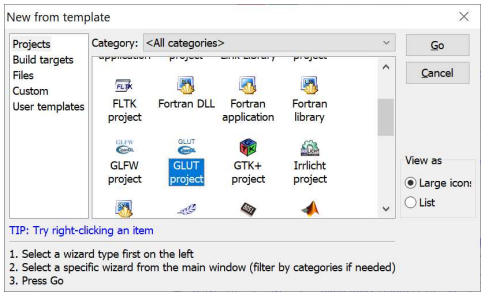


Trong đó:

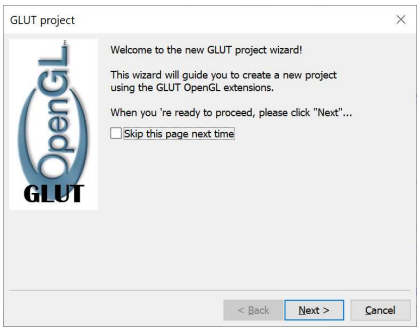
- Thư mục include chứa các tập tin *.h
- Thư mục lib chứa các tập tin *.lib

Bước 3: Chạy CodeBlocks, chọn **File → New → Project**, tiếp đến chọn **GLUT project** và bấm **Go**.



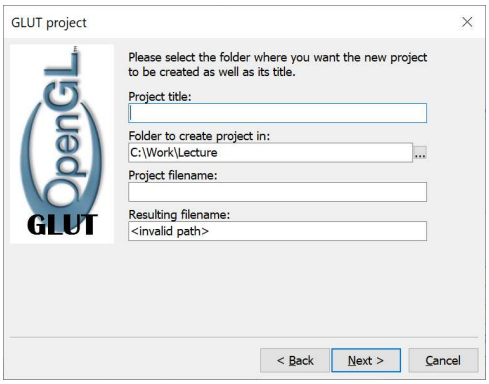


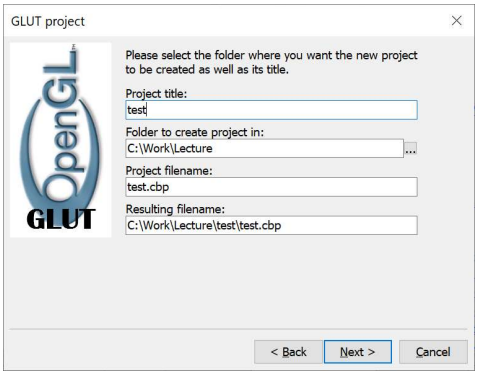
Tiếp đến bấm Next,



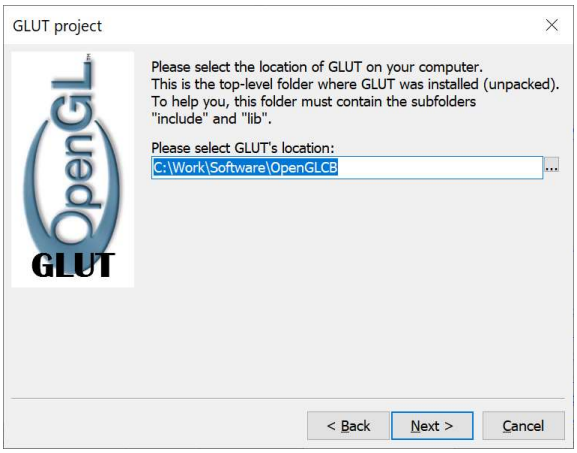
Chọn thư mục chứa project (sắp được tạo) ở **Folder to create project in:**

Tiếp đến gõ tên project ở **Project title:** ,

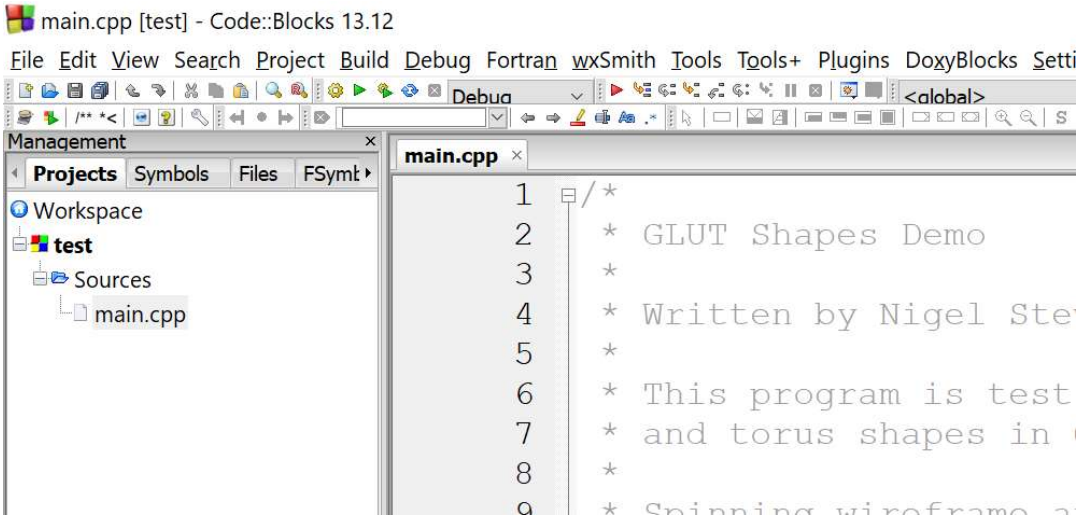




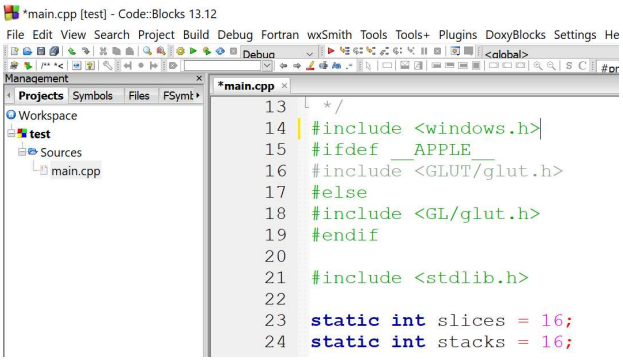
Rồi chọn thư mục chứa thư viện đồ họa, bấm **Next**. Cuối cùng bấm **Finish**.



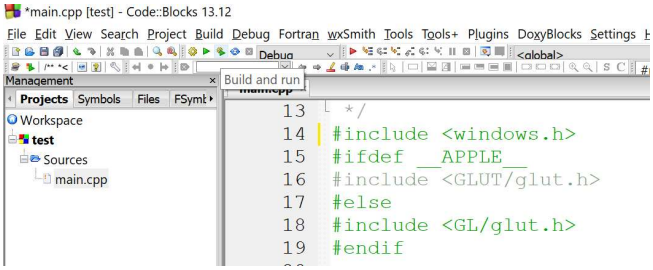
Bước 4: Bấm vào test → Sources → main.cpp, cửa sổ soạn thảo mã nguồn được mở ra



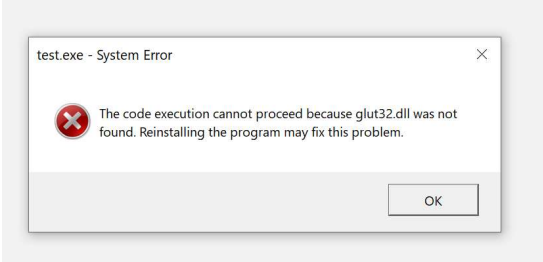
Thêm dòng **#include <windows.h>** vào dòng số 14, trong mã nguồn



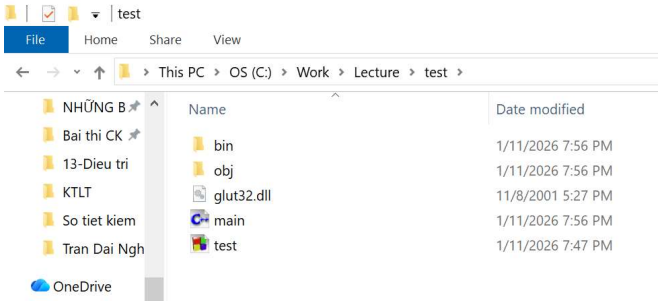
Bấm nút Build and Run



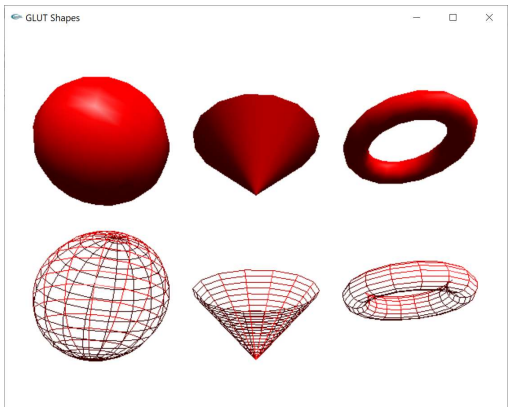
Chương trình xuất hiện lỗi sau (lý do là glut32.dll chưa được copy vào thư mục chứa project)



Copy glut32.dll vào thư mục chứa project



Bấm Build and Run, chương trình chạy ra kết quả như sau:

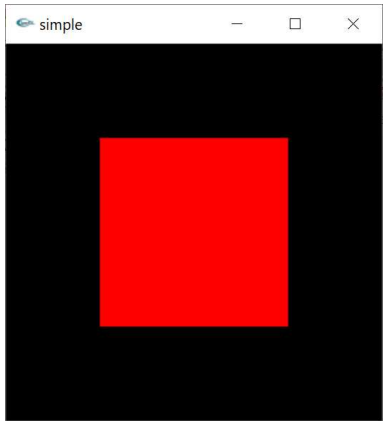


2. MỘT SỐ VÍ DỤ

1) Vẽ hình vuông tĩnh

Xóa toàn bộ nội dung trong tập tin main.cpp, sau đó gõ nội dung sau vào tập tin này và chạy chương trình

```
#include <windows.h>
#include <glut.h>

void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f);
    glBegin(GL_POLYGON);
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
    glEnd();
    glFlush();
}
int main(int argc, char** argv){
    glutCreateWindow("simple");
    glutDisplayFunc(mydisplay);
    glutMainLoop();
}
```



2) Vẽ hình vuông động

```
#include <glut.h>
#include <math.h>
#include <stdio.h>
GLfloat angle;
#define DEG2RAD (3.14159f/180.0f)

void mydisplay(){
 glClear(GL_COLOR_BUFFER_BIT);
 glColor3f(1.0f, 0.0f, 0.0f);

 GLfloat cx, cy;
 cx = 0.5*cos(DEG2RAD*angle); cy = 0.5*sin(DEG2RAD*angle);

 glBegin(GL_POLYGON);
   glVertex2f(cx - 0.5, cy - 0.5);
   glVertex2f(cx - 0.5, cy + 0.5);
   glVertex2f(cx + 0.5, cy + 0.5);
   glVertex2f(cx + 0.5, cy - 0.5);
 glEnd();
 glFlush();
 glutSwapBuffers();

}
void processTimer(int value){
 angle += (GLfloat)value;
 if(angle > 360) angle = angle - 360.0f;
```

```
 glutTimerFunc(100, processTimer, 10);
 glutPostRedisplay();
}

int main(int argc, char** argv){
 angle = 0.0f;
 glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
 glutCreateWindow("simple");
 glutDisplayFunc(mydisplay);
 glutTimerFunc(100, processTimer, 10);
 glutMainLoop();
}
```

3) Vẽ đa diện

```
#include <windows.h>
#include <glut.h>
#define min(a,b) ((a)<(b)?(a):(b))

float tetra_vertices[][3] = {
 {0.0, 0.0, 1.0},
 {0.0, 0.942809, -0.33333},
 {-0.816497, -0.471405, -0.33333},
 {0.816497, -0.471405, -0.33333}
};

void mydisplay(){
 //setup viewwer - camera parameter:
 //i.e., location (eye),
 //direction (from eye to the reference point), and orientation (up vector)
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 gluLookAt(
  1.5, //eyeX
  1.5, //eyeY
  1.5, //eyeX
  0.0, //reference point X
  0.0, //reference point Y
  0.0, //reference point Z
  0.0, //up vector X
  1.0, //up vector Y
  0.0  //up vector Z
  );
```

```
 //clear screen
    glClear(GL_COLOR_BUFFER_BIT);

 //the order of the vertices in a triangle is important!
 glBegin(GL_TRIANGLES);
  //Face 1: defined by vertices: 0, 2, 1; colored: red
  glColor3f(1.0f, 0.0f, 0.0f);
  glVertex3fv(tetra_vertices[0]);
  glVertex3fv(tetra_vertices[2]);
  glVertex3fv(tetra_vertices[1]);

  //Face 2: defined by vertices: 0, 1, 3; colored: red
  glColor3f(0.0f, 1.0f, 0.0f);
  glVertex3fv(tetra_vertices[0]);
  glVertex3fv(tetra_vertices[1]);
  glVertex3fv(tetra_vertices[3]);

  //Face 3: defined by vertices: 0, 3, 2; colored: red
  glColor3f(0.0f, 0.0f, 1.0f);
  glVertex3fv(tetra_vertices[0]);
  glVertex3fv(tetra_vertices[3]);
  glVertex3fv(tetra_vertices[2]);

  //Face 4: defined by vertices: 3, 1, 2; colored: red
  glColor3f(1.0f, 0.0f, 0.0f);
  glVertex3fv(tetra_vertices[3]);
  glVertex3fv(tetra_vertices[1]);
  glVertex3fv(tetra_vertices[2]);

 glEnd();
 glFlush();
 glutSwapBuffers();

}
void reshape(int width, int height){
 //setup viewport
 int size = min(width, height);
 glViewport(0, 0, size, size);
}
void initOpenGL(){
 //setup projection type
 //glFrustrum: define viewing volume
```

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(
  -1.0, //left
  1.0, //right
  -1.0, //bottom
  1.0, //top
  2.0, //near
  10.0 //far
  );
//Default MatrixMode is MODELVIEW
glMatrixMode(GL_MODELVIEW);

//setup background color, or clear color
glClearColor(0.1f, 0.7f, 0.7f, 1.0f);
}

int main(int argc, char** argv){
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
glutCreateWindow("Drawing a Tetrahedron");
initOpenGL();
glutDisplayFunc(mydisplay);
glutReshapeFunc(reshape);
glutMainLoop();
}
```

4) Vẽ đa diện thay đổi vị trí nhìn

```
#include <windows.h>
#include <glut.h>
#include <math.h>
#define min(a,b) ((a)<(b)?(a):(b))
#define DEG2RAD (3.14159f/180.0f)
  float tetra_vertices[][3] = {
  {0.0, 0.0, 1.0},
  {0.0, 0.942809, -0.33333},
  {-0.816497, -0.471405, -0.33333},
  {0.816497, -0.471405, -0.33333}
  };
  GLfloat angle;
void mydisplay(){
  //setup viewwer - camera parameter:
  //i.e., location (eye),
  //direction (from eye to the reference point), and orientation (up vector)
```

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(
1.5*cos(DEG2RAD*angle),
1.5*sin(DEG2RAD*angle),
3.5, //eyeZ
0.0, //reference point X
0.0, //reference point Y
0.0, //reference point Z
0.0, //up vector X
1.0, //up vector Y
0.0
//up vector Z
);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

 //the order of the vertices in a triangle is important!
 glBegin(GL_TRIANGLES);
  //Face 1: defined by vertices: 0, 2, 1; colored: red
  glColor3f(1.0f, 0.0f, 0.0f);
  glVertex3fv(tetra_vertices[0]);
  glVertex3fv(tetra_vertices[2]);
  glVertex3fv(tetra_vertices[1]);

  //Face 2: defined by vertices: 0, 1, 3; colored: red
  glColor3f(0.0f, 1.0f, 0.0f);
  glVertex3fv(tetra_vertices[0]);
  glVertex3fv(tetra_vertices[1]);
  glVertex3fv(tetra_vertices[3]);

  //Face 3: defined by vertices: 0, 3, 2; colored: red
  glColor3f(0.0f, 0.0f, 1.0f);
  glVertex3fv(tetra_vertices[0]);
  glVertex3fv(tetra_vertices[3]);
  glVertex3fv(tetra_vertices[2]);

  //Face 4: defined by vertices: 3, 1, 2; colored: red
  glColor3f(1.0f, 0.0f, 0.0f);
  glVertex3fv(tetra_vertices[3]);
  glVertex3fv(tetra_vertices[1]);
  glVertex3fv(tetra_vertices[2]);
```

```
    glEnd();
    glFlush();
    glutSwapBuffers();

}
void reshape(int width, int height){
    //setup viewport
    int size = min(width, height);
    glViewport(0, 0, size, size);
}

void processTimer(int value){
    angle += (GLfloat)value;
    if(angle > 360.0f) angle -= 360.0f;
    glutTimerFunc(100, processTimer, value);
    glutPostRedisplay();
}
void initOpenGL (){
    //setup projection type
    //glFrustrum: define viewing volume
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(
      -2.0, //left
      2.0, //right
      -2.0, //bottom
      2.0, //top
      2.0, //near
      10.0 //far
      );
    //Default MatrixMode is MODELVIEW
    glMatrixMode(GL_MODELVIEW);
    glEnable(GL_DEPTH_TEST);
}
int main(int argc, char** argv){
   angle = 0.0f;
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
   glutCreateWindow("Drawing a Tetrahedron");
   initOpenGL();
   glutDisplayFunc(mydisplay);
   glutReshapeFunc(reshape);
   glutTimerFunc(100, processTimer, 10);
```

```
    glutMainLoop();
}
```

3. BÀI TẬP

1. Vẽ hình tròn bằng cách xuất ra tập các điểm trên biên của nó. Khoảng cách giữa các điểm tính theo góc là 10 độ.

2. Vẽ hình tròn bằng cách nối các đỉnh ở câu 1.

3. Chia hình tròn ở câu 2. thành các tam giác bằng cách vẽ đường nối với giữa tâm đến các đỉnh nằm trên biên của nó.

4. Chọn 3 tam giác từ tập tam giác ở câu 3, sao cho chúng nằm cân đối trên hình tròn. Tô màu 3 tam giác này. Không vẽ các tam giác khác, để cho 3 tam giác tạo thành cái quạt 3 cánh. Cho cái quạt xoay quanh tâm, cứ sau 100milisec thì quay.