¿En qué lenguaje fue programado Asriel?

Es un compilador programado en Java para generar un ejecutable de código fuente de C++ además de generar código Assembly e Intermediate.

¿Qué características contiene?

Asriel es capaz de realizar lo siguiente:

- Abrir archivos
- Guardar archivos
- Copiar texto seleccionado
- Pegar texto seleccionado
- Cortar texto seleccionado
- Generar ensamblador
- Generar intermedio
- Compilar código fuente
- Correr código fuente

¿Por qué C++?

Por estas simples razones:

- Gran cantidad de librerías.
- Flexibilidad, velocidad y compatibilidad.
- Es usado en muchos softwares actualmente.
- Es un lenguaje independiente de máquina, o que puede ser ejecutado en cualquier arquitectura o kernel.

Requisitos para Asriel:

- Previa instalación de MinGW-W64

https://sourceforge.net/projects/mingw-w64/

- Previa instalación de JRE (Java Runtime Environment)

https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

Nota .- puede que necesite agregar la ruta /bin/ de MinGW a variables del entorno.

¿Qué es MinGW-W64 y por qué se ocupa?

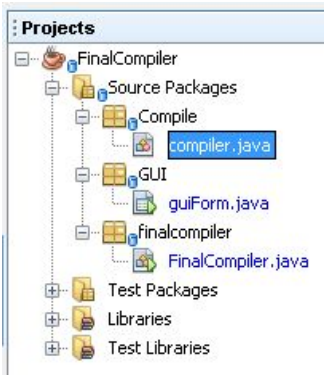
Es un kit completo que te permite compilar y enlazar a la Runtime Library C del componente del sistema operativo Windows en msvcrt.dll. Debido a esto, podemos escribir ejecutables binarios nativos para Windows, no solo en C y C ++, sino también en otros lenguajes.

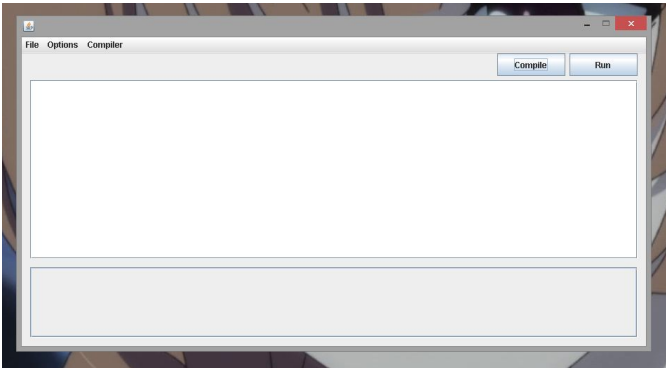
¿Qué es ProcessBuilder?

Es una clase en Java que es usada para crear procesos del sistema operativo. Cada instancia del ProcessBuilder maneja una colección de atributos para el proceso. Con esto, podemos mandar llamar el Runtime MinGW para poder compilar nuestro código C++ y crear un ejecutable.

Explicación del código

Se usó el diseñador de Java para poder graficar una interfaz base con la cual podremos visualizar e interactuar. Nuestro proyecto contiene la siguiente estructura:





Dentro de tal estructura, nos encontramos con un paquete y clase llamada “Final Compiler”. Al abrirla, veremos el siguiente código:

```
public static void main(String[] args) {

    // codigo para ventana
    SwingUtilities.invokeLater(() -> {
        guiForm gui = new guiForm();
        JOptionPane.showMessageDialog(gui, "Asegúrese de tener GCC.exe como Variable de Entorno");
        if (JOptionPane.OK_OPTION == 0) {
            Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
            gui.setVisible(true);
            gui.setResizable(false);
            gui.setLocation(dim.width / 2 - gui.getSize().width / 2, dim.height / 2 - gui.getSize().height / 2);
        }
    ));
}
```

De aquí, mandamos llamar la clase guiForm para poder correr el resto del código. Dentro de nuestra clase guiForm, encontramos las funciones y códigos para todo lo demás.

```
private void ClearActionPerformed(java.awt.event.ActionEvent evt) {

    // Codigo para borrar todo
    try {
        code.setText(" ");
    } catch (Exception exp) {
        JOptionPane.showMessageDialog(null, "Error en -> " + exp.getMessage());
    }
}

private void CutActionPerformed(java.awt.event.ActionEvent evt) {

    // Codigo para cortar
    try {
        code.cut();
    } catch (Exception exp) {
        JOptionPane.showMessageDialog(null, "Error en -> " + exp.getMessage());
    }
}

private void PasteActionPerformed(java.awt.event.ActionEvent evt) {

    // Codigo para pegar
    try {
        code.paste();
    } catch (Exception exp) {
        JOptionPane.showMessageDialog(null, "Error en -> " + exp.getMessage());
    }
}

private void copyAllActionPerformed(java.awt.event.ActionEvent evt) {

    // Codigo para copiar
    try {
        code.copy();
    } catch (Exception exp) {
        JOptionPane.showMessageDialog(null, "Error en -> " + exp.getMessage());
    }
}
```

Mandamos pedir un diálogo para seleccionar un documento, después obtener la respuesta del usuario y comparar si sí se acepta. Al hacerlo, mandamos obtener la ruta del archivo para poder leer su contenido usando la librería File mandando el String ruta a la clase compiler. Finalmente mostramos el texto en el campo de texto.

```
private void openFileActionPerformed(java.awt.event.ActionEvent evt) {

    // codigo para abrir archivo y mostrar texto
    try {
        JFileChooser file = new JFileChooser("f:");
        int r = file.showOpenDialog(null);
        if (r == JFileChooser.APPROVE_OPTION) {
            File path = new File(file.getSelectedFile().getAbsolutePath());
            c = new compiler(path.toString());
            try {
                String s1, s1;
                FileReader fr = new FileReader(path);
                BufferedReader br = new BufferedReader(fr);
                s1 = br.readLine();
                while ((s1 = br.readLine()) != null) {
                    s1 = s1 + "\n" + s1;
                }
                code.setText(s1);
            } catch (FileNotFoundException fileExp) {
                JOptionPane.showMessageDialog(null, "Error en -> " + fileExp.getMessage());
            }
        } else {
            System.out.println("Operation cancelled");
        }
    } catch (IOException expTwo) {
        JOptionPane.showMessageDialog(null, "Error en -> " + expTwo.getMessage());
    }
}
```

Aquí vemos que sí se cumplió la función:

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

int main () {
  // for loop execution
  for( int a = 10; a < 20; a = a + 1 ) {
    cout << "value of a: " << a << endl;
  }

  return 0;
}
```

La instancia a continuación de la clase Compiler, fue creada para poder mover la ruta entre las diferentes funciones de acción de la clase JFrame llamada guiForm.

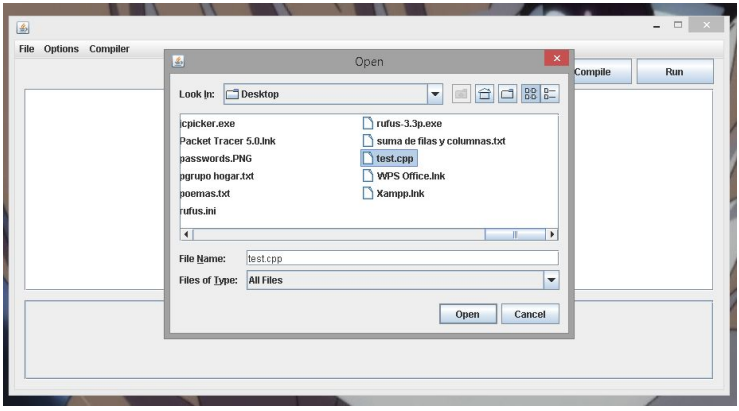
```
public compiler c;
```

A continuación, la clase donde guardamos la ruta. Usamos un método para retornar el valor tipo String. Vemos que también hay un constructor.

```
public class compiler {

    public String filepathC;

    public compiler(String filepath) {
        this.filepathC = filepath;
    }

    public String Compile(){
        return filepathC;
    }
}
```

Después, podemos guardar el archivo editado. Usando la misma implementación del diálogo para seleccionar, podemos elegir la ruta de guardado. Finalmente usamos un FileWriter para sobreescribir o crear nuevo archivo con el contenido del usuario y la extensión que desee.

```
private void saveFileActionPerformed(java.awt.event.ActionEvent evt) {

    // Codigo para guardar el texto en archivo con extension
    try {
        JFileChooser file = new JFileChooser("f:");
        int r = file.showSaveDialog(null);
        if (r == JFileChooser.APPROVE_OPTION) {
            File path = new File(file.getSelectedFile().getAbsolutePath());
            try {
                FileWriter fr = new FileWriter(path, false);
                try (BufferedWriter br = new BufferedWriter(fr)) {
                    br.write(code.getText());
                    br.flush();
                }
            } catch (FileNotFoundException fileExp) {
                JOptionPane.showMessageDialog(null, "Error en -> " + fileExp.getMessage());
            }
        } else {
            System.out.println("Operation cancelled");
        }
    } catch (IOException ioExp) {
        JOptionPane.showMessageDialog(null, "Error en -> " + ioExp.getMessage());
    }
}
```

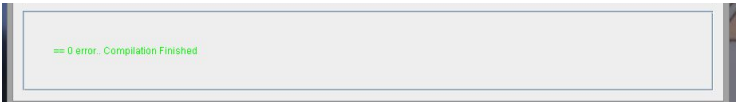


Ahora, vamos a hablar sobre la función de compilar. Aquí usamos un ProcessBuilder para permitirnos usar el terminal del sistema. Aquí están lo que significa cada opción del ProcessBuilder:

- cmd: la terminal del sistema Windows
- /C: para ejecutar la terminal en el fondo
- G++: es la función del MinGW para realizar el Object-Linking con el sistema
- newpath: obtenemos la ruta con el nombre sin la extensión, para usar como archivo origen
- name: el nombre original del archivo
- nameexe: name más la extensión

```
private void CompileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String filepath = c.Compile();
        File file = new File(filepath);
        String path = file.getPath();
        String newpath = path.substring(0, path.lastIndexOf(File.separator));
        String name = file.getName();
        String nameexe = file.getName().substring(0, file.getName().lastIndexOf("."));
        String folder = newpath+"\\";
        System.out.println(folder + "run.bat");
        String exe = folder+nameexe+".exe";

        ProcessBuilder pb;
        try {
            pb = new ProcessBuilder("cmd", "/C", "g++ " + "\"" + newpath + "\\" + name + "\"" + " -o \"" + nameexe+"\"");
            pb.directory(new File(newpath));

            Process p = pb.start();
            p.waitFor();
            int x = p.exitValue();

            if (x == 0) {
                terminal.setForeground(Color.GREEN);
                terminal.setText("          == 0 error.. Compilation Finished");
            } else {
                BufferedReader r = new BufferedReader(new InputStreamReader(p.getErrorStream()));
                String out;
                terminal.setText("");
                while ((out = r.readLine()) != null) {
                    terminal.setForeground(Color.RED);
                    terminal.setText(out + System.getProperty("line.separator"));
                }
            }
        } catch (IOException | InterruptedException ex) {
            JOptionPane.showMessageDialog(null, "Error en -> " + ex.getMessage());
        }
    } catch (HeadlessException exp) {
        JOptionPane.showMessageDialog(null, "Error en -> " + exp.getMessage());
    }
}
```

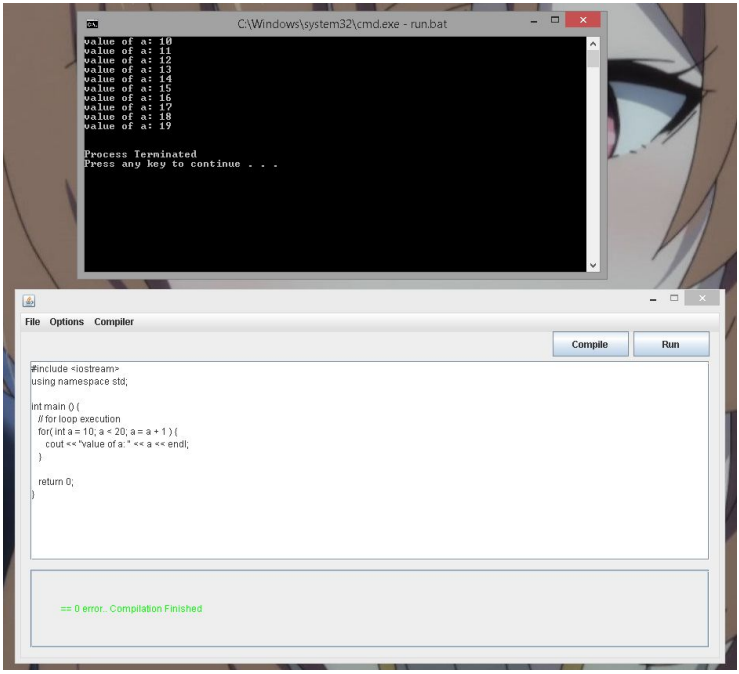
Después mandamos ejecutar el comando, guardamos el exe, y si el código de respuesta es igual a 0 significa que fue un éxito. De lo contrario, leemos lo que nos devolvió de error y lo mostramos.



Ahora, la función de ejecutar.

```
private void RunActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String filepath = c.Compile();
        File file = new File(filepath);
        String path = file.getPath();
        String newpath = path.substring(0, path.lastIndexOf(File.separator));
        System.out.println(newpath);
        String name = file.getName();
        String nameexe = file.getName().substring(0, file.getName().lastIndexOf("."));
        String folder = newpath+"\\";

        ProcessBuilder pb;
        try {
            pb = new ProcessBuilder("cmd", "/C", "g++ " + "\"" + newpath + "\\" + name + "\"" + " -o \"" + nameexe+"\"");
            pb.directory(new File(newpath));
            Process p = pb.start();
            p.waitFor();
            int x = p.exitValue();
            int z = p.exitValue();
            if ( x == 0) {
                //Runtime rt = Runtime.getRuntime();
                try {
                    String co = "@echo off\n" + "\"" +
                                newpath + "\\" + nameexe + ".exe\"\n" + "echo.\n" + "echo.\n" + "echo Process Terminated\n" +
                                "pause\n" +
                                "exit";
                    File dir = new File(newpath + "\\CodeEditor");
                    dir.mkdir();
                    try {
                        File file2 = new File(folder + "\\codeEditor" + "\\run.bat");
                        file2.createNewFile();
                        try (PrintWriter writer = new PrintWriter(file2)) {
                            writer.append(co);
                        }
                        Process p2 = Runtime.getRuntime().exec("cmd /c start run.bat", null, new File(newpath + "\\CodeEditor"));
                    } catch (IOException exxx) {
                    JOptionPane.showMessageDialog(null, "Error en -> " + exxx.getMessage());
                }
                } catch (HeadlessException exx) {
                    JOptionPane.showMessageDialog(null, "Error en -> " + exx.getMessage());
                }
            }
        } catch (IOException | InterruptedException ex) {
            JOptionPane.showMessageDialog(null, "Error en -> " + ex.getMessage());
        }
    } catch (HeadlessException e) {
        JOptionPane.showMessageDialog(null, "Error en -> " + e.getMessage());
    }
}
```

La función de ejecutar código inicial de la misma forma, obteniendo la ruta del archivo ejecutable para poder especificar que queremos ese para compilar. De tal forma que usamos las primeras variables para dar formato y generar los datos que queremos. Después, generamos nuestro ProcessBuilder y corremos el comando generado, avisamos que se ejecutó con éxito para después crear una carpeta llamada “codeEditor” en la cual guardamos un archivo run.bat para correr el ejecutable usando el runtime de MinGW.



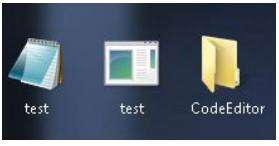
El código de run.bat es el siguiente:

```
@echo off
“ruta de ejecutable”
echo.
echo.
echo Process Terminated
Pause
Exit
```

Con esto, podemos usar Windows para mandar llamar el ejecutable para poder ver la pantalla de terminal que vemos en la imagen anterior. El archivo es automáticamente editado al seleccionar el botón Run usando la variable String co.

Generación de código ensamblador

Al usar MinGW, hay una bandera u opción que te permite detener el generador antes de mandar ejecutar el código ensamblador. Aquí lo obtenemos y generamos un archivo .asm con el que podremos visualizar el ensamblador. Además, contamos con código fuente C/C++ preprocesado. A continuación, los archivos generados en escritorio:



Aquí vemos nuestro método para poder generar el código ensamblador encontrado en CodeEditor:

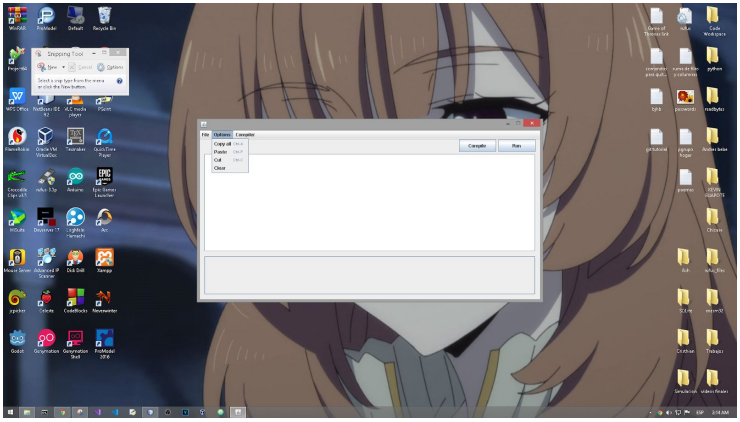
```
private void genASMActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {

        String data = code.getText(), filepath = c.Compile();
        File file = new File(filepath);
        String path = file.getPath();
        String name = file.getName().substring(0, file.getName().lastIndexOf("."));
        String newpath = path.substring(0, path.lastIndexOf(File.separator)) + "\\CodeEditor";
        String save_ASM = newpath + "\\" + name + ".asm";

        if (data != null) {
            ProcessBuilder pb;
            try {
                pb = new ProcessBuilder("cmd", "/C", "g++ " + "-S" + " -o " + save_ASM + " " + filepath);
                pb.directory(new File(newpath));

                Process p = pb.start();
                p.waitFor();
                int x = p.exitValue();

                if (x == 0) {
                    terminal.setForeground(Color.GREEN);
                    terminal.setText("Assembly generated in -> " + save_ASM);
                } else {
                    BufferedReader r = new BufferedReader(new InputStreamReader(p.getErrorStream()));
                    String out;
                    terminal.setText("");
                    while ((out = r.readLine()) != null) {
                        terminal.setForeground(Color.RED);
                        terminal.setText(out + System.getProperty("line.separator"));
                    }
                }
            } catch (IOException | InterruptedException exp) {
                JOptionPane.showMessageDialog(null, "Error en -> " + exp.getMessage());
            }
        } else {
            JOptionPane.showMessageDialog(null, "No contiene código");
        }
    } catch (HeadlessException e) {
        JOptionPane.showMessageDialog(null, "Error en -> " + e.getMessage());
    }
}
```

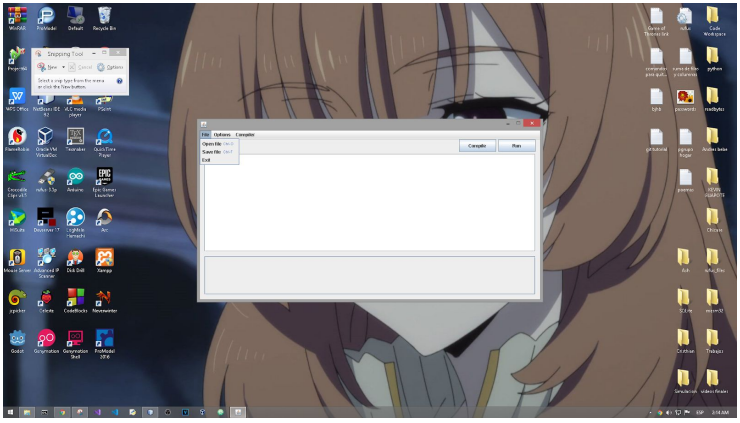
Generación de código intermedio

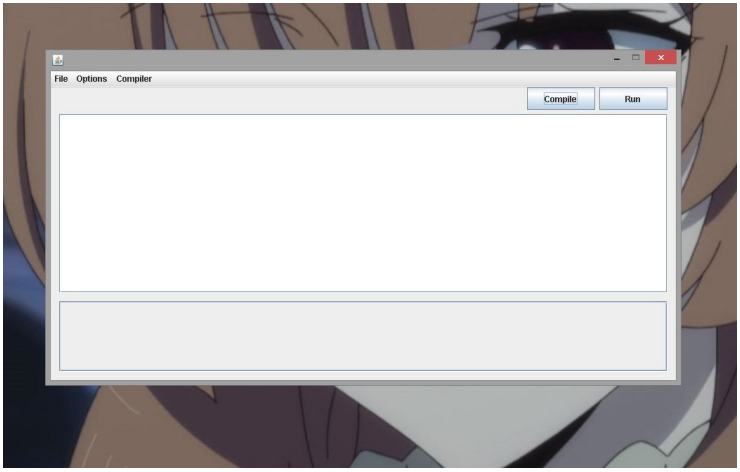
Finalmente, nuestro método de código intermedio:

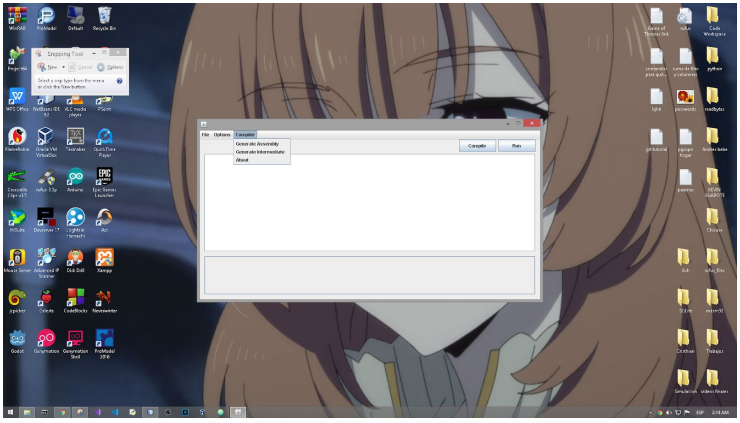
```
private void genIntermediateActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {

        String data = code.getText(), filepath = c.Compile();
        File file = new File(filepath);
        String path = file.getPath();
        String name = file.getName().substring(0, file.getName().lastIndexOf("."));
        String newpath = path.substring(0, path.lastIndexOf(File.separator)) + "\\CodeEditor";
        String save_ASM = newpath + "\\" + name + ".i";

        if (data != null) {
            ProcessBuilder pb;
            try {
                pb = new ProcessBuilder("cmd", "/C", "g++ " + "-S -save-temps" + " -o " + save_ASM + " " + filepath);
                pb.directory(new File(newpath));

                Process p = pb.start();
                p.waitFor();
                int x = p.exitValue();

                if (x == 0) {
                    terminal.setForeground(Color.GREEN);
                    terminal.setText("Intermediate Code generated in -> " + save_ASM);
                } else {
                    BufferedReader r = new BufferedReader(new InputStreamReader(p.getErrorStream()));
                    String out;
                    terminal.setText("");
                    while ((out = r.readLine()) != null) {
                        terminal.setForeground(Color.RED);
                        terminal.setText(out + System.getProperty("line.separator"));
                    }
                }
            } catch (IOException | InterruptedException exp) {
                JOptionPane.showMessageDialog(null, "Error en -> " + exp.getMessage());
            }
        } else {
            JOptionPane.showMessageDialog(null, "No contiene código");
        }
    } catch (HeadlessException e) {
        JOptionPane.showMessageDialog(null, "Error en -> " + e.getMessage());
    }
}
```

Pruebas de funcionalidad

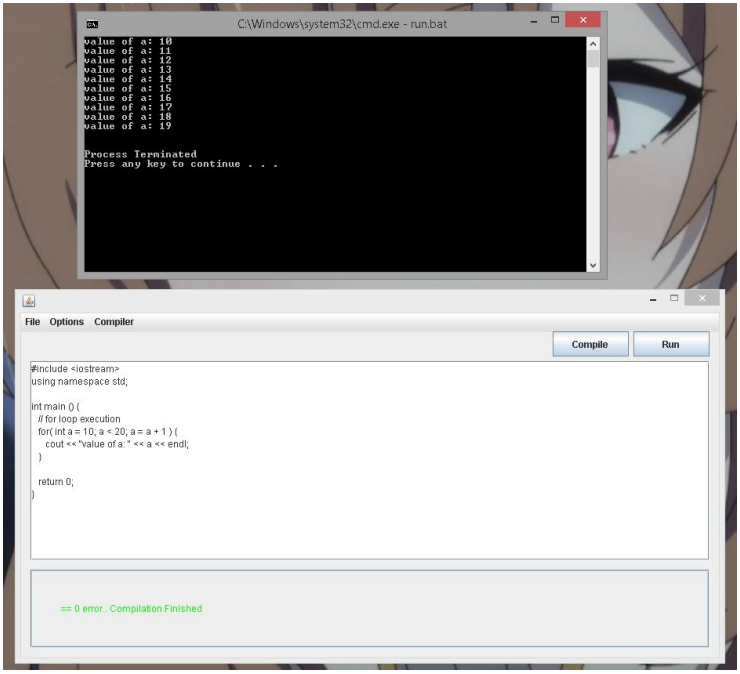


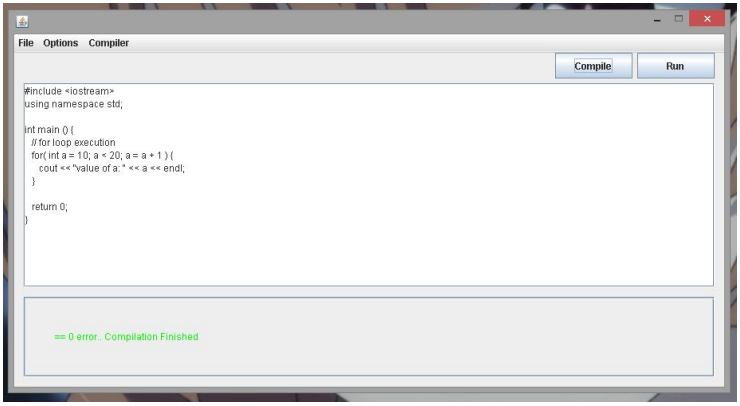


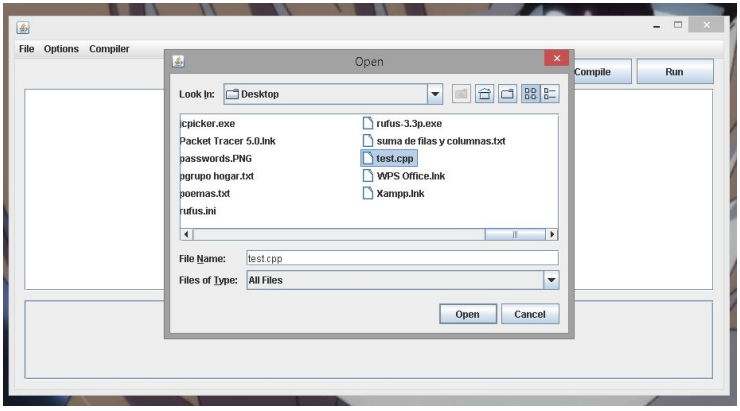

File
Options
Compiler
Compile
Run




Message
Asegúrese de tener GCC.exe como Variable de Entorno
OK


C:\Windows\system32\cmd.exe - run.bat
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
Process Terminated
Press any key to continue . . .
File Options Compiler
Compile
Run
#include <iostream>
using namespace std;
int main () {
// for loop execution
for(int a = 10; a < 20; a = a + 1) {
cout << "value of a: " << a << endl;
}
return 0;
}
== 0 error.. Compilation Finished


File Options Compiler
Compile
Run
#include <iostream>
using namespace std;
int main () {
// for loop execution
for(int a = 10; a < 20; a = a + 1) {
cout << "value of a:" << a << endl;
}
return 0;
}
== 0 error, Compilation Finished


File Options Compiler
Compile
Run
Open
Look In:
Desktop
icpicker.exe
Packet Tracer 5.0.lnk
passwords.PNG
pgrupo hogar.txt
poemas.txt
rufus.ini
rufus-3.3p.exe
suma de filas y columnas.txt
test.cpp
WPS Office.lnk
Xampp.lnk
File Name:
test.cpp
Files of Type:
All Files
Open
Cancel