



INSTITUTO POLITECNICO NACIONAL



ESCUELA SUPERIOR DE INGENIERIA
MECANICA Y ELECTRICA

INGENIERIA EN COMPUTACION

INTEGRANTES:

MONTES SIERRA PABLO.

RODRIGUEZ APARICIO YAIR.

SANTAMARIA GUERRERO DIEGO.

GRUPO: 5CX32

コンパイラ

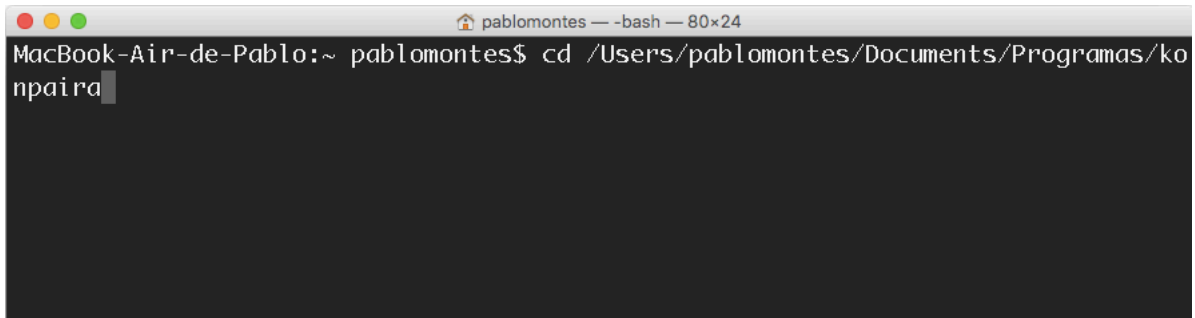
(Konpaira)

índice

Manual de Usuario.....	3
Ejemplos en lenguaje kompaira.....	12
Manual Técnico.....	18
Diagramas de uso UML.....	51
Diccionario de Datos.....	52
Conclusiones.....	53

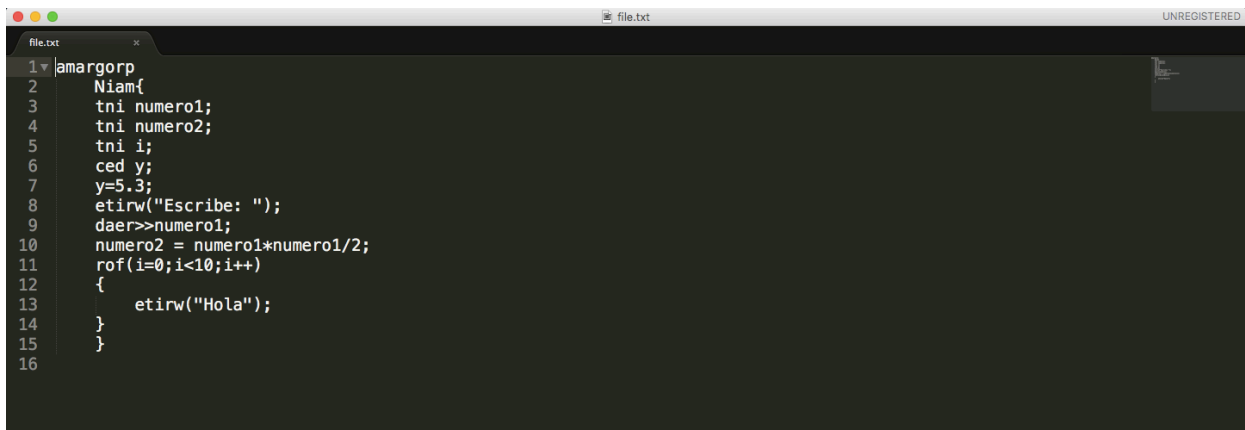
Manual de Usuario

Imagen 1:



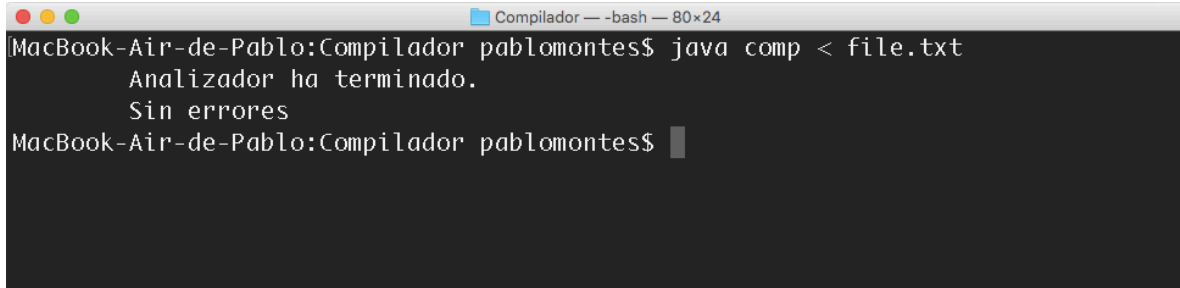
Se ingresa a terminal desde la computadora, estando allí, se escribe, “**cd ruta**”, donde ruta es la ruta donde se encuentra la carpeta del compilador, se escribe sin las comillas, como se muestra en la imagen 1

Imagen 2:



Creamos un archivo con el nombre: **file** y la extensión **.txt** en la misma ruta donde se encuentra la carpeta del compilador. Ahí se escribe el código a realizar, un ejemplo esta en la imagen 2.

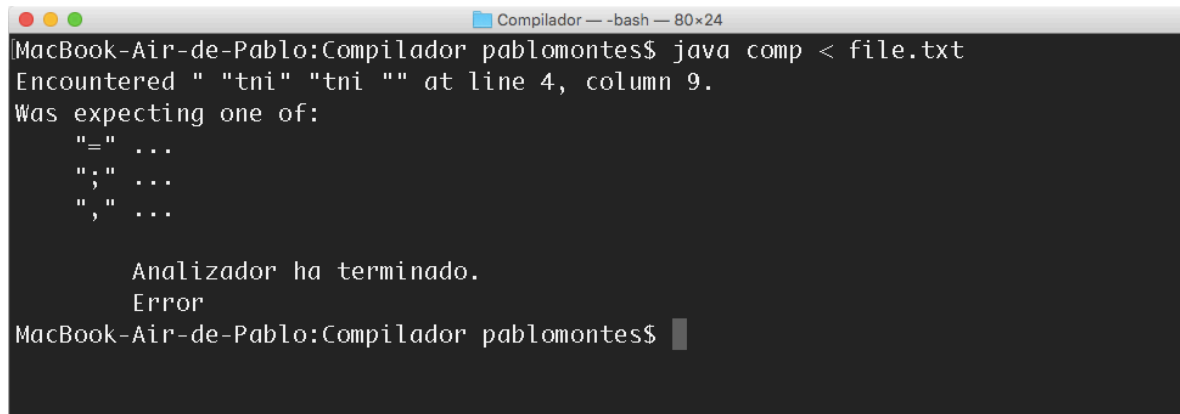
Imagen 3:

A terminal window titled 'Compilador - -bash - 80x24' on a MacBook-Air-de-Pablo. The prompt is 'MacBook-Air-de-Pablo:Compilador pablomontes\$'. The user enters 'java comp < file.txt'. The output is 'Analizador ha terminado.' followed by 'Sin errores' on the next line. The prompt returns to 'MacBook-Air-de-Pablo:Compilador pablomontes\$' with a cursor.

```
MacBook-Air-de-Pablo:Compilador pablomontes$ java comp < file.txt
Analizador ha terminado.
Sin errores
MacBook-Air-de-Pablo:Compilador pablomontes$
```

Una vez en la ruta de la carpeta en terminal, escribir el comando “**java comp < file.txt**”, donde file es el nombre con la extensión donde se guardo el código fuente, en el ejemplo el archivo fuente se guardo con el nombre: file y con la extensión .txt, se escribe sin comillas toda la línea de comandos, como se muestra en la imagen 2.

Si el código no tiene ningún error léxico, sintáctico y/o semántico, se mostrara en pantalla de terminal: “**Analizador ha terminado. Sin errores**”

A terminal window titled 'Compilador - -bash - 80x24' on a MacBook-Air-de-Pablo. The prompt is 'MacBook-Air-de-Pablo:Compilador pablomontes\$'. The user enters 'java comp < file.txt'. The output is 'Encountered " "tni" "tni "' at line 4, column 9.' followed by 'Was expecting one of:' and a list of expected tokens: '"="' ...', '";" ...', and '" ," ...'. Then it says 'Analizador ha terminado.' followed by 'Error' on the next line. The prompt returns to 'MacBook-Air-de-Pablo:Compilador pablomontes\$' with a cursor.

```
MacBook-Air-de-Pablo:Compilador pablomontes$ java comp < file.txt
Encountered " "tni" "tni "' at line 4, column 9.
Was expecting one of:
    "=" ...
    ";" ...
    "," ...

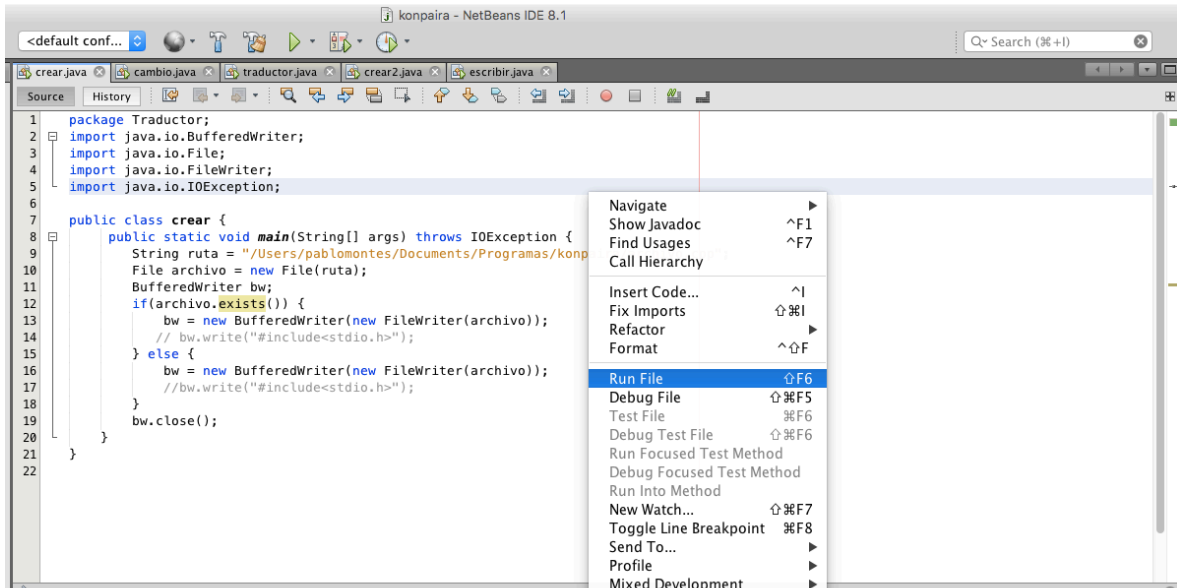
Analizador ha terminado.
Error
MacBook-Air-de-Pablo:Compilador pablomontes$
```

Imagen 4:

Al realizar el paso anterior, si el archivo fuente, contiene algún error, ya sea léxico, sintáctico y/o semántico, se mostrara en pantalla de terminal: “**Analizador ha terminado. Error**”.

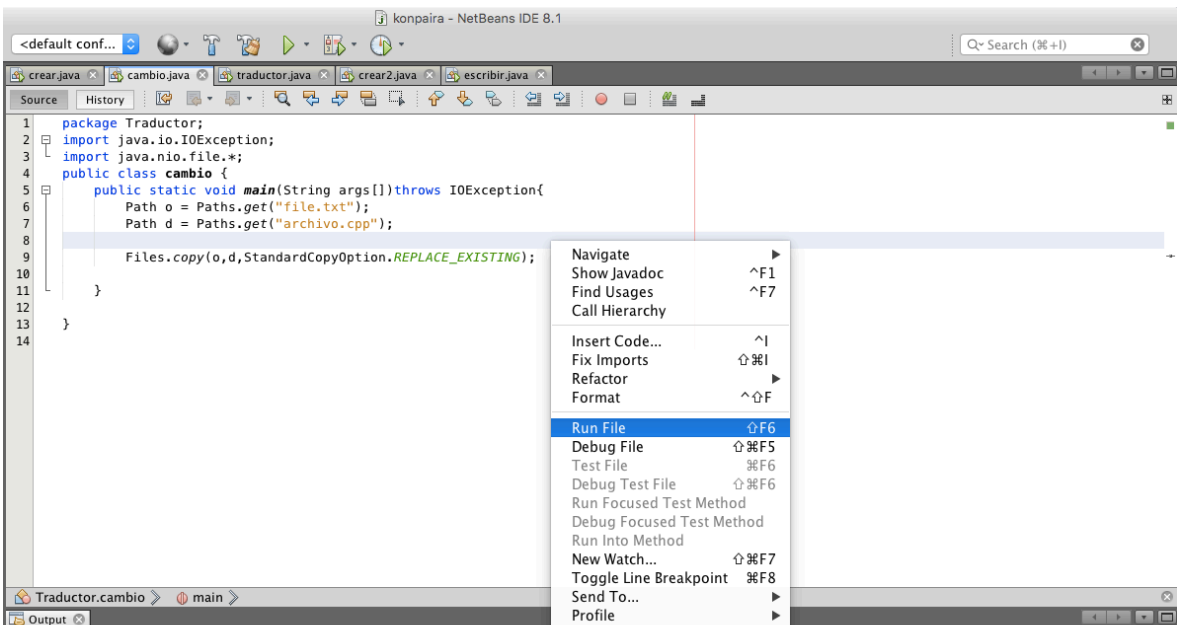
En caso de que suceda esto, remitirse a ejemplos de código y ver como se declara, se hace una función y/o ejecuta una sentencia.

Imagen 5:



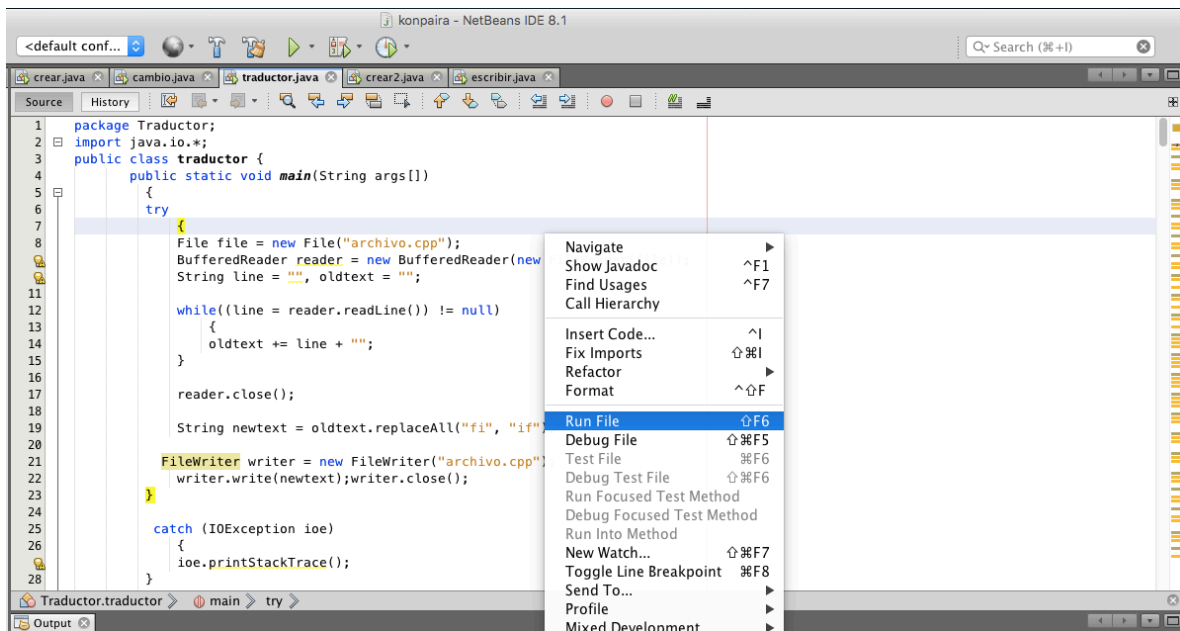
Abrimos el proyecto del compilador, nos dirigimos a la pestaña crear, en el entorno, damos click derecho y click sobre Run File.

Imagen 6:



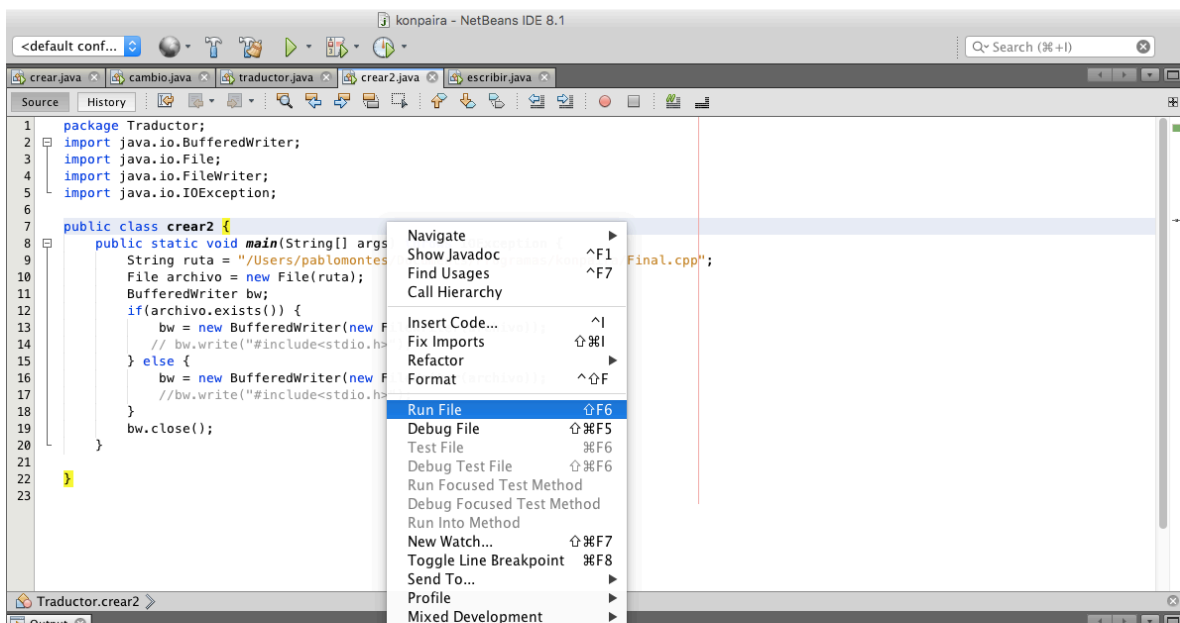
Abrimos el proyecto del compilador, nos dirigimos a la pestaña cambio, en el entorno, damos click derecho y click sobre Run File.

Imagen 7:



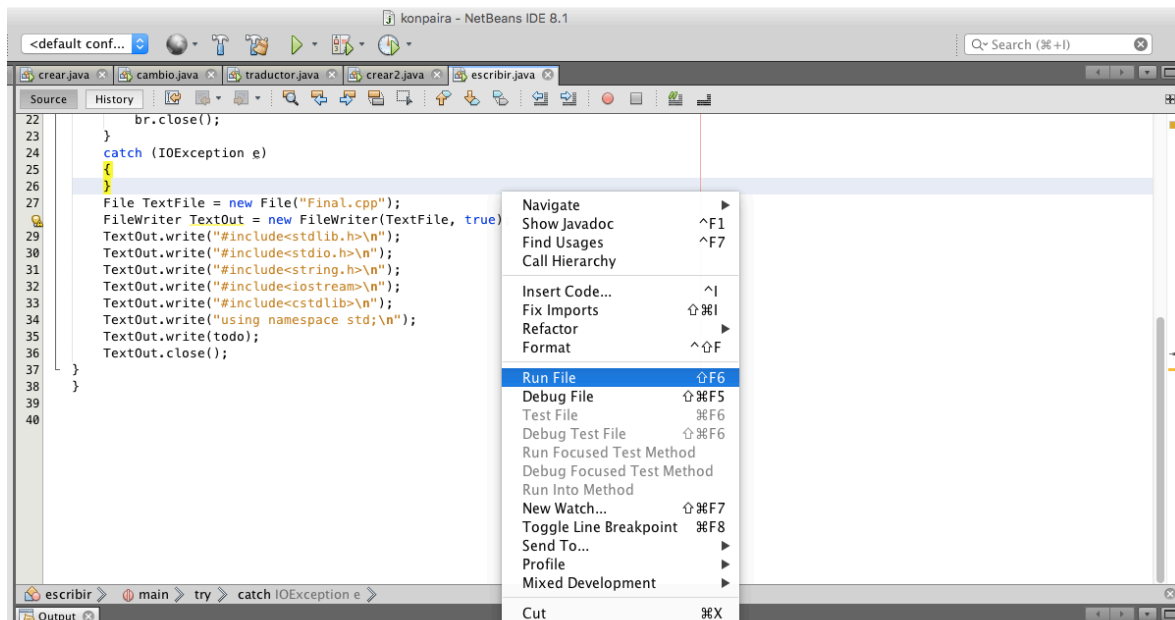
Abrimos el proyecto del compilador, nos dirigimos a la pestaña traductor, en el entorno, damos click derecho y click sobre Run File.

Imagen 8:



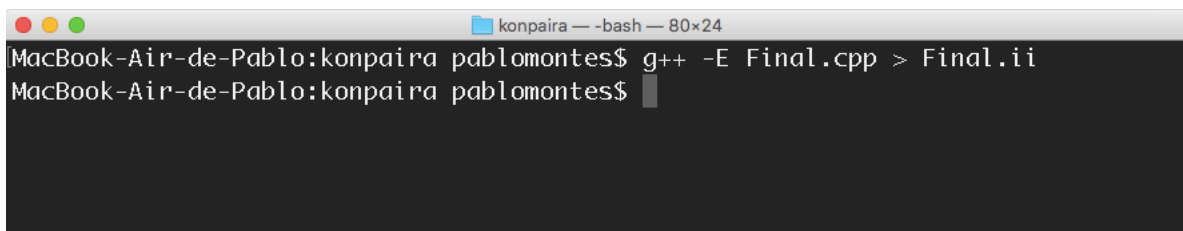
Abrimos el proyecto del compilador, nos dirigimos a la pestaña crear2, en el entorno, damos click derecho y click sobre Run File.

Imagen 9:



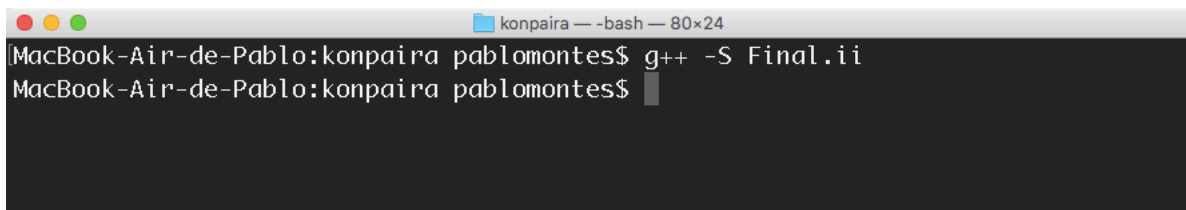
Abrimos el proyecto del compilador, nos dirigimos a la pestaña escribir, en el entorno, damos click derecho y click sobre Run File.

Imagen 10:



En terminal, sobre la misma ruta donde se encuentra el compilador, escribimos la línea de comando: “**g++ -E Final.cpp > Final.ii**”, sin comillas.

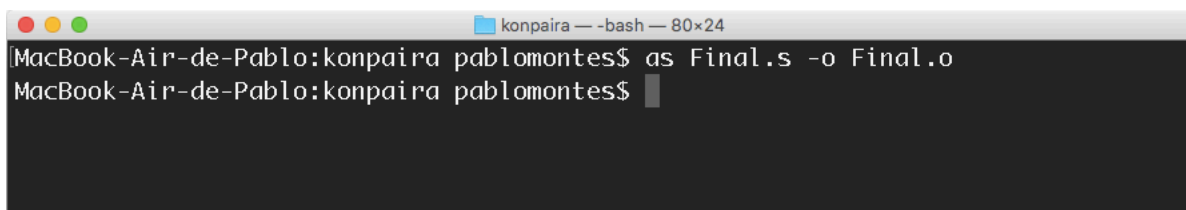
Imagen 11:

A terminal window titled 'konpaira — -bash — 80x24' on a Mac. The prompt is 'MacBook-Air-de-Pablo:konpaira pablomontes\$'. The command 'g++ -S Final.ii' has been entered and executed. The prompt is now 'MacBook-Air-de-Pablo:konpaira pablomontes\$' with a cursor at the end.

```
MacBook-Air-de-Pablo:konpaira pablomontes$ g++ -S Final.ii
MacBook-Air-de-Pablo:konpaira pablomontes$
```

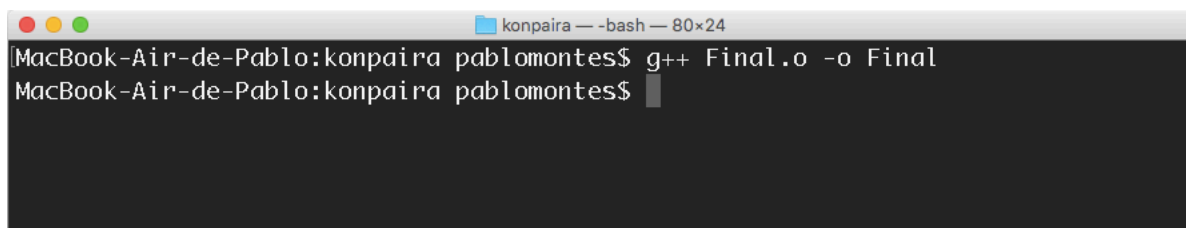
En terminal, sobre la misma ruta donde se encuentra el compilador, escribimos la línea de comando: “**g++ -S Final.ii**”, sin comillas.

Imagen 12:

A terminal window titled 'konpaira — -bash — 80x24' on a Mac. The prompt is 'MacBook-Air-de-Pablo:konpaira pablomontes\$'. The command 'as Final.s -o Final.o' has been entered and executed. The prompt is now 'MacBook-Air-de-Pablo:konpaira pablomontes\$' with a cursor at the end.

```
MacBook-Air-de-Pablo:konpaira pablomontes$ as Final.s -o Final.o
MacBook-Air-de-Pablo:konpaira pablomontes$
```

En terminal, sobre la misma ruta donde se encuentra el compilador, escribimos la línea de comando: “**as Final.s -o Final.o** ”, sin comillas.

A terminal window titled 'konpaira — -bash — 80x24' on a Mac. The prompt is 'MacBook-Air-de-Pablo:konpaira pablomontes\$'. The command 'g++ Final.o -o Final' has been entered and executed. The prompt is now 'MacBook-Air-de-Pablo:konpaira pablomontes\$' with a cursor at the end.

```
MacBook-Air-de-Pablo:konpaira pablomontes$ g++ Final.o -o Final
MacBook-Air-de-Pablo:konpaira pablomontes$
```

Imagen 13:

En terminal, sobre la misma ruta donde se encuentra el compilador, escribimos la línea de comando: “**g++ Final.o -o Final** ”, sin comillas.

Imagen 14:



En la carpeta donde se encuentra creado el compilador, estos serán los nuevos archivos generados desde terminal.

Imagen 15:

A screenshot of a code editor window titled 'Final.ii'. The editor shows a list of preprocessor directives for a C++ compilation. The code is as follows:

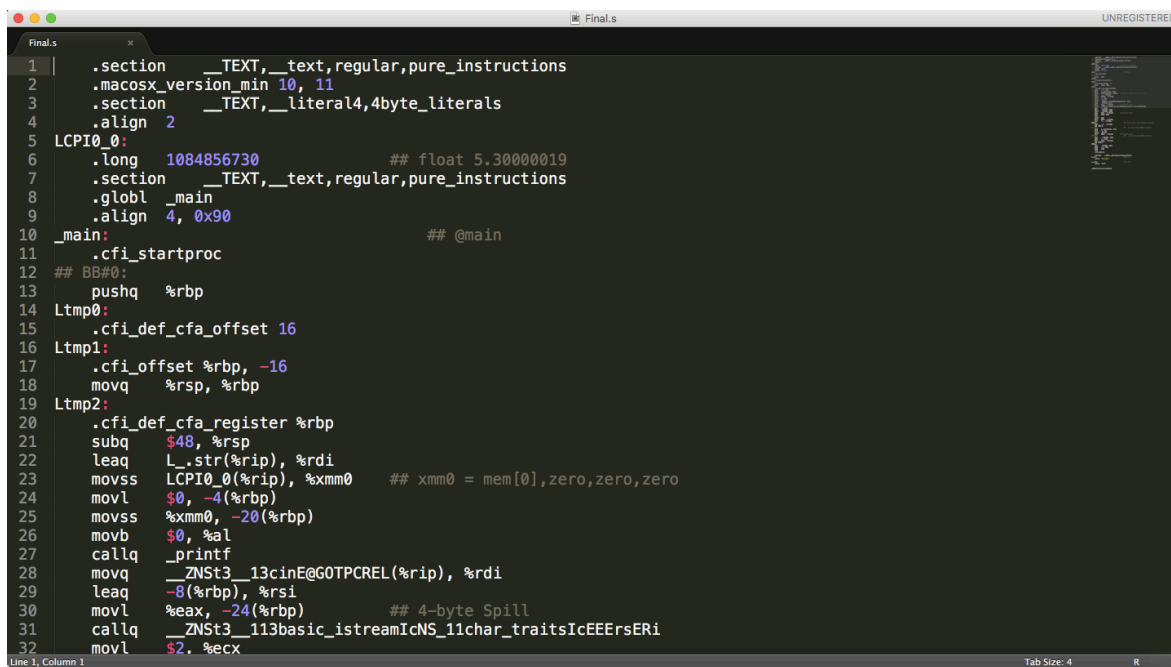
```
1 # 1 "Final.cpp"
2 # 1 "<built-in>" 1
3 # 1 "<built-in>" 3
4 # 332 "<built-in>" 3
5 # 1 "<command line>" 1
6 # 1 "<built-in>" 2
7 # 1 "Final.cpp" 2
8 # 1 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/stdlib.h" 1 3 4
9 # 61 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/stdlib.h" 3 4
10 # 1 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/Availability.h" 1 3 4
11 # 164 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/Availability.h" 3 4
12 # 1 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/AvailabilityInternal.h" 1 3 4
13 # 165 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/Availability.h" 2 3 4
14 # 62 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/stdlib.h" 2 3 4
15
16 # 1 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/_types.h" 1 3 4
17 # 27 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/_types.h" 3 4
18 # 1 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/sys/_types.h" 1 3 4
19 # 32 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/sys/_types.h" 3 4
20 # 1 "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.
sdk/usr/include/sys/cdefs.h" 1 3 4
```

The editor interface includes a status bar at the bottom showing 'Line 1, Column 1' and 'Spaces: 4 Plain Text'.

Si se abre el archivo Final.ii, se observa las líneas de código intermedio

コンパイラ

Imagen 16:



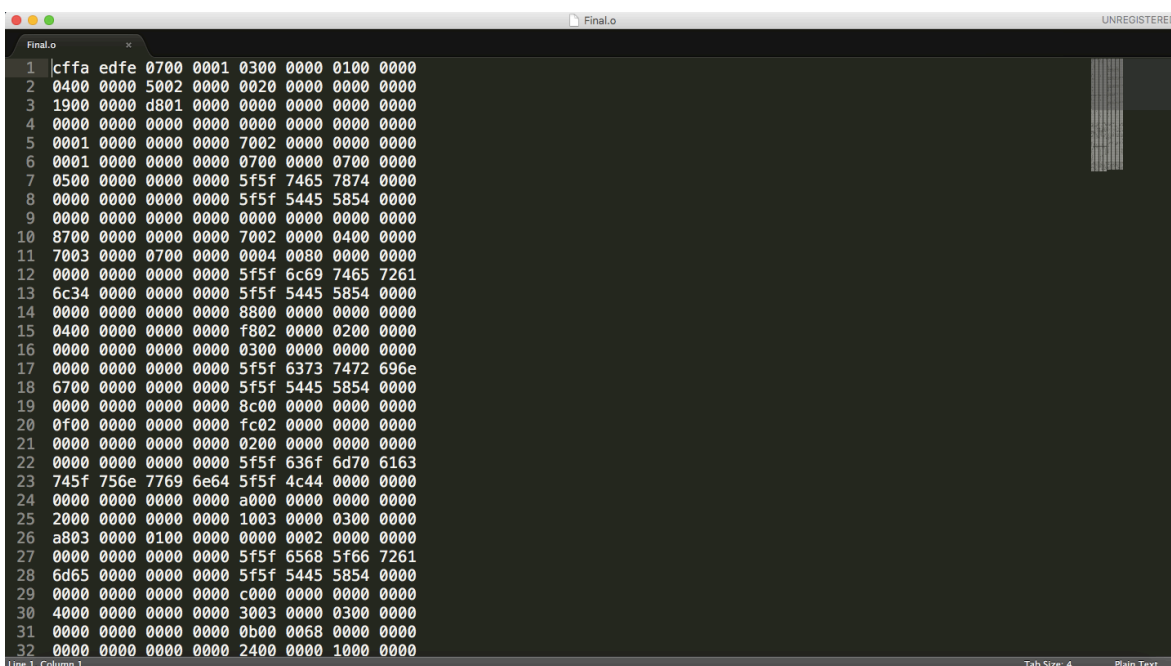
```

1 | .section __TEXT,__text,regular,pure_instructions
2 | .macosx_version_min 10, 11
3 | .section __TEXT,__literal4,4byte_literals
4 | .align 2
5 | LCPI0_0:
6 | .long 1084856730 ## float 5.30000019
7 | .section __TEXT,__text,regular,pure_instructions
8 | .globl _main
9 | .align 4, 0x90
10 | _main: ## @main
11 | .cfi_startproc
12 | ## BB#0:
13 | pushq %rbp
14 | Ltmp0:
15 | .cfi_def_cfa_offset 16
16 | Ltmp1:
17 | .cfi_offset %rbp, -16
18 | movq %rsp, %rbp
19 | Ltmp2:
20 | .cfi_def_cfa_register %rbp
21 | subq $48, %rsp
22 | leaq L_.str(%rip), %rdi
23 | movss LCPI0_0(%rip), %xmm0 ## xmm0 = mem[0],zero,zero,zero
24 | movl $0, -4(%rbp)
25 | movss %xmm0, -20(%rbp)
26 | movb $0, %al
27 | callq _printf
28 | movq __ZNSt3_13cinE@GOTPCREL(%rip), %rdi
29 | leaq -8(%rbp), %rsi
30 | movl %eax, -24(%rbp) ## 4-byte Spill
31 | callq __ZNSt3_11basic_istreamIcNS_11char_traitsIcEEEErsEri
32 | movl $2, %ecx

```

Si se abre el archivo Final.s, se observa las líneas de código ensamblador.

Imagen 17:



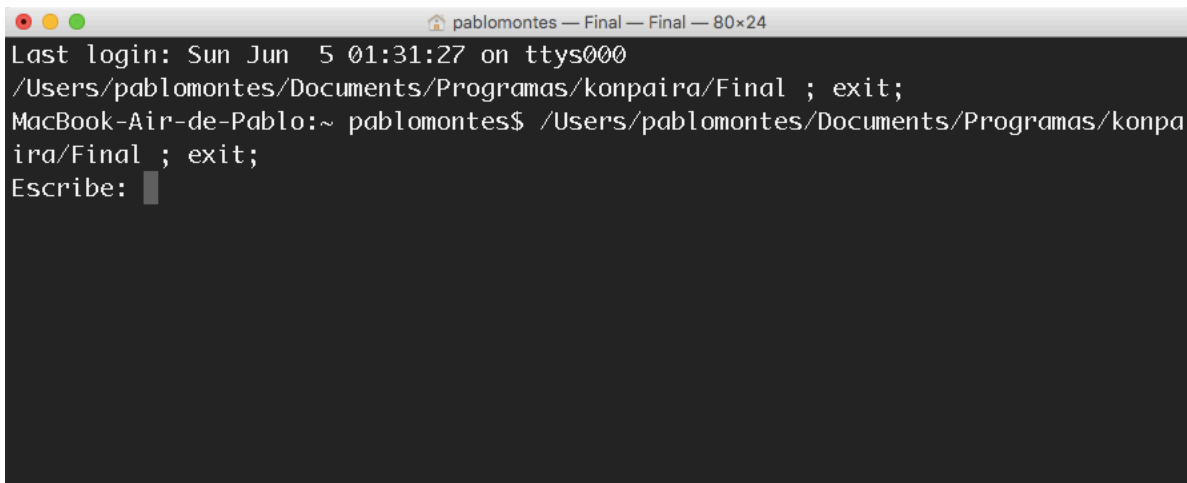
```

1 | cffa edfe 0700 0001 0300 0000 0100 0000
2 | 0400 0000 5002 0000 0020 0000 0000 0000
3 | 1900 0000 d801 0000 0000 0000 0000 0000
4 | 0000 0000 0000 0000 0000 0000 0000 0000
5 | 0001 0000 0000 0000 7002 0000 0000 0000
6 | 0001 0000 0000 0000 0700 0000 0700 0000
7 | 0500 0000 0000 0000 5f5f 7465 7874 0000
8 | 0000 0000 0000 0000 5f5f 5445 5854 0000
9 | 0000 0000 0000 0000 0000 0000 0000 0000
10 | 8700 0000 0000 0000 7002 0000 0400 0000
11 | 7003 0000 0700 0000 0004 0080 0000 0000
12 | 0000 0000 0000 0000 5f5f 6c69 7465 7261
13 | 6c34 0000 0000 0000 5f5f 5445 5854 0000
14 | 0000 0000 0000 0000 8800 0000 0000 0000
15 | 0400 0000 0000 0000 f802 0000 0200 0000
16 | 0000 0000 0000 0000 0300 0000 0000 0000
17 | 0000 0000 0000 0000 5f5f 6373 7472 696e
18 | 6700 0000 0000 0000 5f5f 5445 5854 0000
19 | 0000 0000 0000 0000 8c00 0000 0000 0000
20 | 0f00 0000 0000 0000 fc02 0000 0000 0000
21 | 0000 0000 0000 0000 0200 0000 0000 0000
22 | 0000 0000 0000 0000 5f5f 636f 6d70 6163
23 | 745f 756e 7769 6e64 5f5f 4c44 0000 0000
24 | 0000 0000 0000 0000 a000 0000 0000 0000
25 | 2000 0000 0000 0000 1003 0000 0300 0000
26 | a803 0000 0100 0000 0000 0002 0000 0000
27 | 0000 0000 0000 0000 5f5f 6568 5f66 7261
28 | 6d65 0000 0000 0000 5f5f 5445 5854 0000
29 | 0000 0000 0000 0000 c000 0000 0000 0000
30 | 4000 0000 0000 0000 3003 0000 0300 0000
31 | 0000 0000 0000 0000 0b00 0068 0000 0000
32 | 0000 0000 0000 0000 2400 0000 1000 0000

```

Si se abre el archivo Final.o, se observa las líneas de código objeto.

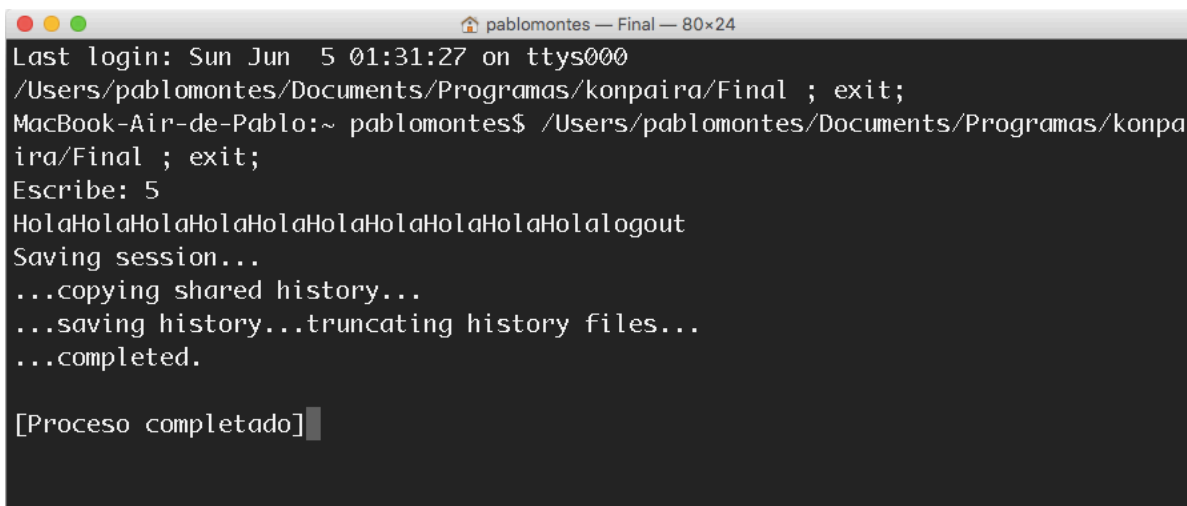
Imagen 18:

A screenshot of a macOS Terminal window. The title bar at the top reads "pablomontes — Final — Final — 80x24". The terminal text shows a successful login, followed by the execution of a command to run a program located at "/Users/pablomontes/Documents/Programas/konpaira/Final". The prompt "Escribe:" is displayed with a cursor, indicating the program is waiting for input.

```
pablomontes — Final — Final — 80x24
Last login: Sun Jun  5 01:31:27 on ttys000
/Users/pablomontes/Documents/Programas/konpaira/Final ; exit;
MacBook-Air-de-Pablo:~ pablomontes$ /Users/pablomontes/Documents/Programas/konpaira/Final ; exit;
Escribe: 
```

Si se abre el archivo Final.exec, se abre Terminal donde ya este el programa corriendo, como se observa en la imagen 18, pide ingresar un valor.

Imagen 19:

A screenshot of the same macOS Terminal window as in Image 18. The user has entered the number "5" in response to the "Escribe:" prompt. The program then outputs a series of "Hola" messages, followed by "logout", "Saving session...", and progress messages for copying shared history and saving history files. The terminal concludes with "[Proceso completado]" and a cursor.

```
pablomontes — Final — 80x24
Last login: Sun Jun  5 01:31:27 on ttys000
/Users/pablomontes/Documents/Programas/konpaira/Final ; exit;
MacBook-Air-de-Pablo:~ pablomontes$ /Users/pablomontes/Documents/Programas/konpaira/Final ; exit;
Escribe: 5
HolaHolaHolaHolaHolaHolaHolaHolaHolaHolalogout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Proceso completado] 
```

En la imagen 19, ya se ve el programa que ha terminado de ejecutarse.

Ejemplos en lenguaje kompaira

Programa 1->prueba con rof

amargorp

```
Niam{  
    rof(tni valor=0;valor<10;valor++){  
        etirw ("prueba");  
    }  
}
```

Programa 2->prueba con fi

amargorp

```
Niam{  
    tni val1 = 3;  
    tni val2 = 5;  
    fi(val1<val2){  
        etirw("exito");  
    }  
}
```

Programa 3->prueba con fi y rof

amargorp

```
Niam{  
    tni val1 = 3;  
    tni val2 = 5;
```

```

    tni ciclo;

    fi(val1<val2){
        rof(ciclo=0;ciclo<5;ciclo++){
            etirw("esto es un ciclo");
        }
    }
}

```

Programa 4->prueba fi-esle

amargorp

```

    Niam{
        tni val1 = 6;
        tni val2 = 5;
        fi(val1<val2){
            etirw("incorrecto");
        }
        esle{
            etirw("correcto");
        }
    }
}

```

Programa 5->prueba for, fi-else

amargorp

```

    Niam{
        tni val1 = 6;

```

```

tni val2 = 5;

tni ciclo;

fi(val1<val2){
    etirw("incorrecto");
}

esle{
    rof(ciclo=0;ciclo<val2;ciclo++){
        etirw("hola");
    }
}
}

```

Programa 6->prueba fi-for

amargorp

```

Niam{
    tni val1 = 6;
    tni val2 = 5;
    tni ciclo;
    rof(ciclo=0;ciclo<val1;ciclo++){
        fi(ciclo==val2){
            etirw("condicion");
        }
    }
}
}

```

Programa 7 for

amargorp

```
Niam{  
    tni ciclo;  
    rof(ciclo=0;ciclo<5;ciclo++){  
        etirw("ciclo");  
    }  
}
```

Programa 8 entrada de datos

amargorp

```
Niam{  
    tni val1;  
    tni val2;  
    tni ciclo;  
  
    etirw("Hacer un ciclo");  
    etirw("From: ");
```

```

daer>>val1;

etirw("to: ");

daer>>val2;

rof(ciclo=val1;ciclo<val2;ciclo++){
    etirw("ciclo");
}
}

```

Programa 9 entrada de datos

amargorp

```

Niam{
    tni val1;
    tni val2;
    tni resul;

    etirw("Valor 1: ");
    daer>>val1;
    etirw("Valor 2: ");
    daer>>val2;

    resul= val1 + val2;

    fi(resul<10){
        etirw("la suma de valor 1 y valor 2 es menor a 10");
    }
}

```



```
}  
esle{  
    etirw("la suma de valor 1 y valor 2 es mayor a 10");  
}  
}
```

Manual Técnico

Analisis lexico, sintactico y semántico

```
// Palabra reservada de
javacc donde inicializa
el Parseo y la clase se
llama igual al archivo

PARSER_BEGIN(comp
)
```

```
import java.io.*;

class comp // se crea
la clase comp
```

```
{

// se declara una
bandera para marcar
Error o No Error

static int bandera = 0;
```

```
// Se genera el
Main y la excepcion de
Parse
```

```
public static
void main( String[]
args )throws
ParseException,
Exception
```

```
{

try
```

```
{

//
Nombre de clase y del
objeto. Parametros
System.in todo lo que
entra en el sistema
```

```
comp
analizador = new
comp( System.in ) ;
```

```
//
Programa metodo
principal de la sintaxis
```

```
analizador.Progr
ama();
```

```
//
mensaje de ha
terminado el
analizador
```

```
System.out.print
ln("\tAnalizador ha
terminado.");
```

```
//
Bandera donde indica
que no hay errores
```

```

        bandera = 1;

        if(bandera==1){
            //
            mensaje de no hay
            errores

            System.out.print
            ln("\tSin errores");

        }

        //
        excepcion Parse
        exception

        catch(ParseExce
        ption e)

        {

            System.out.print
            ln(e.getMessage());
            //imprime mensaje

            System.out.print
            ln("\tAnalizador ha
            terminado."); //
            mensaje de ha
            terminado el
            analizador

```

```

        System.out.print
        ln("\tError"); // mensaje
        de Error

    }

}

// se cierra el Parse y
se pasa como
argumento comp,
nombre de la clase
PARSER_END(comp)

// se inicializa el
metodo token para
operadores
aritmeticos

TOKEN :

{

    <ASIGNACION
    : "=">

    | <PLUS : "+" >

    | <MINUS:"- ">

    |
    <MULTIPLY:"*">

    | <DIVIDE:"/">

    | <INCR: "++" >

    | <DECR: "--" >

```

```

    }

    // se inicializa el
    metodo token para
    palabras reservadas

    TOKEN:

    {
        <PUBLIC:
        "cilbup">

        | <PRIVATE:
        "etavirp">

        | <STATIC:
        "citats">

        | <VOID: "diov">

        | <MAIN:
        "Niam">

        | <PROGRAMA:
        "amargorp" >

        | <IF: "fi" >

        | <ELSE:"esle" >

        | <ELSEIF:"esle
        fi" >

        | <FOR:"rof" >

        |
        <SWITCH:"hctiws" >

        | <CASE:"esac"
        >

        |
        <BREAK:"kaerb" >

```

```

    |
    <DEFAULT:"luafed" >

    | <DO:"od" >

    |
    <WHILE:"elihw" >

    | <WRITE:"etirw"
    >

    | <READ:"daer"
    >

    }

    // se inicializa el
    metodo token para
    operadores
    matematicos

    TOKEN:

    {
        <LPAREN: "(" >

        | <RPAREN: ")"
        >

        | <LBRACE: "{"
        >

        | <RBRACE: "}"
        >

        | <LBRACKET:
        "[" >

        | <RBRACKET:
        "]" >

        | <SEMICOLON:
        ";" >

```

```

| <COMMA: "," >

| <DOT: "." >

|
<DOUBLECOMMA:
"\>

}

```

// se inicializa el
metodo token para
operadores logicos

TOKEN:

```

{

<EQ: "==" >

| <LE: "<=" >

| <MN: "<" >

| <GE: ">=" >

| <GR: ">" >

| <NE: "!=" >

| <OR: "||" >

| <AND: "&&">

}

```

// se inicializa el
metodo token para
operadores

TOKEN:

```

{

```

```

<INT:"tni"> //44

```

```

| <DEC: "ced">
//45

```

```

| <CHR: "rahc">
//46

```

```

| <STR: "gnirts">
//47

```

```

| <NUMBER :
(["0"-"9"])+> //48

```

```

| <IDENTIFIER :
["a"-"z","A"-"Z"](["a"-"
z","A"-"Z","0"-"
9","_"])*> //49

```

```

| <DECIMAL :
(["0"-"9"])+["."](["0"-"
9"])+ > //50

```

```

| <CADENA :
<DOUBLECOMMA>["a
"-"z","A"-"Z","0"-"9","
"](["a"-"z","A"-"Z","0"-"
9","
",";",";",".")*<DOUBLE
COMMA>> //51

```

```

| <CARAC :
""["a"-"z","A"-"Z","0"-"
9"]"" > //52

```

```

|
<DOUBLEPOINT : ":">

}

```

// Todos estos
argumentos no los
analiza

<pre> SKIP : { " " "\n" "\r" "\r\n" "\t" } // Estructura del programa principal void Programa() : { { <PROGRAMA>B loque()<EOF> } // Bloque de como declarar variables globales void Bloque(): { { LOOKAHEAD(2) VariablesGlobales() Principal() } // Gramatica de variables </pre>	<pre> void VariablesGlobales(): { { (<PRIVATE> <P UBLIC>)(<INT> <DEC> <STR> <CHR>) <IDENTIFIER>(<COMM A><IDENTIFIER>)* <SEMICOLON> Bloque() } // Cuerpo del programa el main void Principal(): { TokenAsignacio nes.SetTables(); } { <MAIN> <LBRACE>Sente ncias()<RBRACE> } // Gramatica de variables locales </pre>
---	---

<pre> void VariablesLocales(): { int td; Token var; } { (// Identifica que tipo de dato tiene (TiposDatos()) { td = token.kind; } var = <IDENTIFIER> { TokenAsignacio nes.InsertarSimbolo(v ar, td); } // Veificacion de varibel con asigancion [VariablesAsign acion(var)] (</pre>	<pre> <COMMA> var = <IDENTIFIER> { TokenAsignacio nes.InsertarSimbolo(v ar, td); } [VariablesAsign acion(var)])* <SEMICOLON>) (VS()) } // Asignacion de variables void VariablesAsignacion(T oken v1): { Token v2; Token v3; String res; </pre>
--	---

```

        boolean imp =
            false;

        }

        {

            // Validacion de
            asigancion

            <ASIGNACION>

            (TiposAsignacio
            nes())

            {

                v2 =
                token;

                res =
                TokenAsignaciones.ch
                eckAsing(v1, v2);

                if(res != "
                ")

                {

                    System.out.print
                    ln(res);

                    imp
                    = true;

                }

            }

            // Validacion de
            asigancion a traves de
            operadores
            aritmeticos

```

```

        (OpAritmetico())

        TiposAsignaciones()

        {

            v3 =
            token;

            res =
            TokenAsignaciones.ch
            eckAsing(v1, v3);

            if(res != "
            " && !imp)

            {

                System.out.print
                ln(res);

            }

        })*

        }

        // Gramatica de
        variables o sentencias

        void VS():

        {

        }

        LOOKAHEAD(3)
        VariablesLocales() |
        Sentencias()

        }

```



```

// Metodo de
// sentencias
void Sentencias():
{
{
(

VariablesLocales()
| Sentencialf()
| SentenciaFor()
| SentenciaDo()

|
LOOKAHEAD(2)SentenciaAsignacion()<SEMI
COLON>{TokenAsignaciones.segunda = 0;}

|
SentenciaWrite()

|
SentenciaRead()<SEMI
COLON>

|
SentenciaSwitch()

)*

}

//Sentencia IF

```

```

void Sentencialf():
{
{
<IF><LPAREN>
A() <RPAREN>
<LBRACE>
Sentencias()
<RBRACE> (Sino())*

}

void Sino():
{
{
<ELSE><LBRAC
E> Sentencias()
<RBRACE> | <ELSEIF>
<LPAREN> A()
<RPAREN> <LBRACE>
Sentencias()
<RBRACE>

}

// Metodo para
comparar dos valores

void A():
{
{

Comparaciones()
([(<AND>|<OR>)]
Comparaciones())*

```

<pre> } //Fin sentencia if // Funcion de comparacion void Comparaciones(): {} { Valor()Operador es()Valor() } void Valor(): {} { LOOKAHEAD(2) (<IDENTIFIER> <NUMBER>) Expresion() } void Expresion(): {} { LOOKAHEAD(2)(<NUM BER> <IDENTIFIER>) (<IDENTIFIER> <NUMBER>) OpAritmetico() </pre>	<pre> (<IDENTIFIER> <NUMB ER>) } // Metodo para los operadores void Operadores(): {} { <EQ> <LE> <GE> <NE> <GR> <MN> } // Metodo para operadores aritmeticos int OpAritmetico(): {} { (<PLUS> {return 1;}) (<MINUS> {return 1;}) </pre>
--	---

```

| (<MULTIPLY>
{return 1;})

| (<DIVIDE>
{return 1;})

| ({return 0;})

{return 0;}

}

```

**// Metodo para tipo de
datos**

void TiposDatos():

```

{}

{

<INT>

|<DEC>

|<STR>

|<CHR>

}

```

//Sentencia FOR

void SentenciaFor():

```

{}

{

```

```

<FOR>

<LPAREN>(Declaracio
nUnaVariable())|Senten
ciaAsignacion())<SEMI
COLON>

Comparaciones()
<SEMICOLON>

SentenciaAsignacion()
<RPAREN> <LBRACE>

Sentencias()
<RBRACE>

}

```

**// Declaracion de
variable**

**void
DeclaracionUnaVariabl
e():**

```

{

int td;

Token var;

}

```

```

{

(TiposDatos())

```

```

{

```

```

td =
token.kind;

```

```

}

```

```

var =
<IDENTIFIER>

```

```

{

```

```

TokenAsignaciones.InsertarSimbolo(
    var, td);
    }

```

```

[VariablesAsignacion(var)]
    }

```

```

//Sentencia DO
void SentenciaDo():
    {}
    {
        <DO>
        <LBRACE>Sentencias(
        ) <RBRACE> <WHILE>
        <LPAREN>
        Comparaciones()
        <RPAREN> |
        SentenciaWhile()

    }

```

```

// Sentencia While
void SentenciaWhile():
    {}
    {
        <WHILE>
        <LPAREN>
        Comparaciones()

```

```

<RPAREN> <LBRACE>
    Sentencias()
    <RBRACE>
    }

```

```

//Sentencia
ASIGNACION

void
SentenciaAsignacion()
:
{
    Token v1;
    Token v2;
    Token v3;
    int aux;
    String res;
    boolean imp =
    false;
    }
    {
        v1 = <IDENTIFIER>

```

```

        (<ASIGNACION>
        (TiposAsignaciones()))
        {
            v2 = token;

```

```

        res =
TokenAsignaciones.ch
eckAsing(v1, v2);

```

```

        if(res != " ")

```

```

        {

```

```

        System.out.print
        ln(res);

```

```

            imp =
            true;

```

```

        }

```

```

    }

```

```

(LOOKAHEAD(2)OpAri
tmetico())

```

```

TiposAsignaciones()

```

```

{

```

```

    v3 = token;

```

```

        res =
TokenAsignaciones.ch
eckAsing(v1, v3);

```

```

        if(res != " " &&
        !imp)

```

```

        {

```

```

        System.out.print
        ln(res);

```

```

        }

```

```

    })*

```

```

    |

```

```

    (<INCR>|<DECR>)

```

```

    {

```

```

        res =
TokenAsignaciones.ch
eckVariable(v1);

```

```

        if(res != " ")

```

```

        System.out.print
        ln(res);

```

```

    })

```

```

//SentenciaAsignacion
ya trae <SEMICOLON>
en el metodo
Sentencias()

```

```

}

```

```

// Sentencia tipo de
asignaciones

```

```

void
TiposAsignaciones():

```

```

{

```

```

{

```

```

    <IDENTIFIER>

```

```

    | <NUMBER>

```

<DECIMAL>	{
<CADENA>	<READ><GR><G
<CARAC>	R><IDENTIFIER>
}	}
//Sentencia WRITE	// Sentencia Switch
(printf)	void
void SentenciaWrite():	SentenciaSwitch():
{	{
{	{
<WRITE>	<SWITCH><LPA
<LPAREN>	REN><IDENTIFIER><R
(Expresion())(<PLUS><	PAREN><LBRACE>(<
CADENA>)* <CADENA	CASE>(<IDENTIFIER>
>(<PLUS>Expresion())*	<CADENA> <CARAC>
)* <RPAREN>	<NUMBER> <DECIMAL
<SEMICOLON>	>)<DOUBLEPOINT>(Se
}	ntenciaAsignacion())<S
	EMICOLON>)+
	<BREAK><SEMICOLO
	N>)+[<DEFAUL><DOU
//Sentencia READ	BLEPOINT>(Sentencia
(scanf)	Asignacion())<SEMI
void SentenciaRead():	COLON><BREAK><SEMI
{	COLON>]<RBRACE>
}	}

Clase crear: Esta clase genera el archivo .cpp vacío

```
package Traductor;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

//*****Esta clase genera
el archivo .cpp vacío

public class crear {

    public static void
main(String[] args) throws
IOException {

    String ruta =
"/Users/pablomontes/Docume
nts/Programas/konpaira/archi
vo.cpp";

    File archivo = new
File(ruta);

    BufferedWriter bw;

    if(archivo.exists()) {

        bw = new
BufferedWriter(new
FileWriter(archivo));

    } else {

        bw = new
BufferedWriter(new
FileWriter(archivo));

    }

    bw.close();

}
```

Clase Cambio: Esta clase copia el archivo original de nuestro lenguaje.

```
package Traductor;

import java.io.IOException;

import java.nio.file.*;

public class cambio {    /////*****Copia el archivo fuente tal cual
                        de nuestro lenguaje

    public static void main(String args[])throws IOException{

        Path o = Paths.get("file.txt");

        Path d = Paths.get("archivo.cpp");

        Files.copy(o,d,StandardCopyOption.REPLACE_EXISTING);

    }

}
```


Clase Traductor: Esta clase se encarga de convertir nuestro lenguaje de kompira a lenguaje C.

```
package Traductor;

import java.io.*;

//***** Aqui tenemos la clase
// traductor, esta traduce
// nuestro lenguaje a lenguaje c
//*****

public class traductor {

    public static void
    main(String args[])

    {

        try

        {

            File file = new
            File("archivo.cpp"); // leer
            archivo origen

            BufferedReader reader
            = new BufferedReader(new
            FileReader(file));

            String line = "", oldtext
            = "";

            while((line =
            reader.readLine()) != null)

            {

                oldtext += line + "";

            }

            reader.close(); // Deja
            de leer

            // remplazar la palabra
            en el archivo

            //Remplazar la linea en
            el archivo

            String newtext =
            oldtext.replaceAll("fi", "if");
```

<pre> FileWriter writer = new FileWriter("archivo.cpp"); // escribir archivo origen writer.write(newtext);writer.close(); } catch (IOException ioe) { ioe.printStackTrace(); } try { File file = new File("archivo.cpp"); // escribir archivo origen BufferedReader reader = new BufferedReader(new FileReader(file)); String line2 = "", oldtext2 = ""; </pre>	<pre> while((line2 = reader.readLine()) != null) { oldtext2 += line2 + ""; } reader.close(); String newtext = oldtext2.replaceAll("elihw", "while"); //reemplaza la palabra FileWriter writer = new FileWriter("archivo.cpp"); // escribir archivo origen writer.write(newtext);writer.close(); } catch (IOException ioe) { ioe.printStackTrace(); } try { </pre>
---	--

```

        File file = new
File("archivo.cpp"); // escribir
        archivo origen

        BufferedReader reader
= new BufferedReader(new
        FileReader(file));

        String line3 = "",
        oldtext3 = "";

        while((line3 =
reader.readLine()) != null)
        {
            oldtext3 += line3 +
            "";
        }

        reader.close();

        String newtext =
oldtext3.replaceAll("esle",
        "else"); //

        FileWriter writer = new
FileWriter("archivo.cpp"); //
        escribir archivo origen

writer.write(newtext);writer.clo
        se();

    }

```

```

        catch (IOException ioe)
        {
            ioe.printStackTrace();
        }
        try
        {
            File file = new
File("archivo.cpp"); // leer
            archivo origen

            BufferedReader reader
= new BufferedReader(new
            FileReader(file));

            String line4 = "",
            oldtext4 = "";

            while((line4 =
reader.readLine()) != null)
            {
                oldtext4 += line4 +
                "";
            }

            reader.close();

            String newtext =
oldtext4.replaceAll("rof",
"for"); // remplazar la palabra

```

```

        FileWriter writer = new
FileWriter("archivo.cpp"); //
        escribir archivo origen

```

```

writer.write(newtext);writer.close();
        }

```

```

        catch (IOException ioe)
        {

```

```

            ioe.printStackTrace();
        }

```

```

        try

```

```

        {

```

```

            File file = new
File("archivo.cpp"); // leer
            archivo origen

```

```

            BufferedReader reader
= new BufferedReader(new
            FileReader(file));

```

```

            String line5 = "",
            oldtext5 = "";

```

```

            while((line5 =
reader.readLine()) != null)
            {

```

```

                oldtext5 += line5 +
                "";
            }

```

```

            reader.close();

```

```

            String newtext =
oldtext5.replaceAll("etirw",
            "printf"); // remplaza la
            palabra

```

```

            FileWriter writer = new
FileWriter("archivo.cpp"); //
            escribir archivo origen

```

```

writer.write(newtext);writer.close();
        }

```

```

        catch (IOException ioe)
        {

```

```

            ioe.printStackTrace();
        }

```

```

        try

```

```

        {

```

```

            File file = new
File("archivo.cpp"); // leer
            archivo origen

```

```

        BufferedReader reader
= new BufferedReader(new
        FileReader(file));

        String line6 = "",
        oldtext6 = "";

        while((line6 =
        reader.readLine()) != null)
        {
            oldtext6 += line6 +
            "";
        }

        reader.close();

        String newtext =
        oldtext6.replaceAll("od",
        "do"); // remplaza la palabra

        FileWriter writer = new
        FileWriter("archivo.cpp"); //
        leer archivo origen

        writer.write(newtext);writer.clo
        se();

    }

    catch (IOException ioe)

    {

```

```

        ioe.printStackTrace();
    }

    try
    {

        File file = new
        File("archivo.cpp"); // leer
        archivo origen

        BufferedReader reader
        = new BufferedReader(new
        FileReader(file));

        String line7 = "",
        oldtext7 = "";

        while((line7 =
        reader.readLine()) != null)
        {
            oldtext7 += line7 +
            "";
        }

        reader.close();

        String newtext =
        oldtext7.replaceAll("diov",
        "void"); // reemplaza la
        palabra

```

<pre> FileWriter writer = new FileWriter("archivo.cpp"); // leer archivo origen writer.write(newtext);writer.close(); } catch (IOException ioe) { ioe.printStackTrace(); } try { File file = new File("archivo.cpp"); // leer archivo origen BufferedReader reader = new BufferedReader(new FileReader(file)); String line8 = "", oldtext8 = ""; while((line8 = reader.readLine()) != null) { </pre>	<pre> oldtext8 += line8 + ""; } reader.close(); String newtext = oldtext8.replaceAll("cilbup", "public"); // remplaza la palabra FileWriter writer = new FileWriter("archivo.cpp"); // leer archivo origen writer.write(newtext);writer.close(); } catch (IOException ioe) { ioe.printStackTrace(); } try { File file = new File("archivo.cpp"); // leer archivo origen </pre>
--	---

```

        BufferedReader reader
= new BufferedReader(new
        FileReader(file));

        String line9 = "",
        oldtext9 = "";

        while((line9 =
        reader.readLine()) != null)
        {
            oldtext9 += line9 +
            "";
        }

        reader.close();

        String newtext =
        oldtext9.replaceAll("etavirp",
        "private"); // remplaza la
        palabra

        FileWriter writer = new
        FileWriter("archivo.cpp"); //
        leer archivo origen

        writer.write(newtext);writer.clo
        se();
    }

    catch (IOException ioe)
    {

```

```

        ioe.printStackTrace();
    }

    try
    {

        File file = new
        File("archivo.cpp"); // leer
        archivo origen

        BufferedReader reader
        = new BufferedReader(new
        FileReader(file));

        String line9 = "",
        oldtext9 = "";

        while((line9 =
        reader.readLine()) != null)
        {
            oldtext9 += line9 +
            "";
        }

        reader.close();

        String newtext =
        oldtext9.replaceAll("citats",
        "static"); // remplaza la
        palabra

```

<pre> FileWriter writer = new FileWriter("archivo.cpp"); // leer archivo origen writer.write(newtext);writer.close(); } catch (IOException ioe) { ioe.printStackTrace(); } try { File file = new File("archivo.cpp"); // leer archivo origen BufferedReader reader = new BufferedReader(new FileReader(file)); String line10 = "", oldtext10 = ""; while((line10 = reader.readLine()) != null) { </pre>	<pre> oldtext10 += line10 + ""; } reader.close(); String newtext = oldtext10.replaceAll("Niam", "int main()"); // remplaza la palabra FileWriter writer = new FileWriter("archivo.cpp"); // escribe archivo origen writer.write(newtext);writer.close(); } catch (IOException ioe) { ioe.printStackTrace(); } try { File file = new File("archivo.cpp"); // leer archivo origen </pre>
---	---


```

        BufferedReader reader
= new BufferedReader(new
        FileReader(file));

        String line11 = "",
        oldtext11 = "";

        while((line11 =
        reader.readLine()) != null)
        {
            oldtext11 += line11
            + "";
        }

        reader.close();

        String newtext =
        oldtext11.replaceAll("hctiws",
        "switch");

        FileWriter writer = new
        FileWriter("archivo.cpp"); //
        escribe archivo origen

        writer.write(newtext);writer.clo
        se();
    }
    catch (IOException ioe)
    {

```

```

        ioe.printStackTrace();
    }
    try
    {
        File file = new
        File("archivo.cpp"); // leer
        archivo origen

        BufferedReader reader
        = new BufferedReader(new
        FileReader(file));

        String line12 = "",
        oldtext12 = "";

        while((line12 =
        reader.readLine()) != null)
        {
            oldtext12 += line12
            + "";
        }

        reader.close();

        String newtext =
        oldtext12.replaceAll("esac",
        "case"); // remplaza la palabra

```

<pre> FileWriter writer = new FileWriter("archivo.cpp"); // escribe archivo origen writer.write(newtext);writer.close(); } catch (IOException ioe) { ioe.printStackTrace(); } try { File file = new File("archivo.cpp"); // leer archivo origen BufferedReader reader = new BufferedReader(new FileReader(file)); String line13 = "", oldtext13 = ""; while((line13 = reader.readLine()) != null) { </pre>	<pre> oldtext13 += line13 + ""; } reader.close(); String newtext = oldtext13.replaceAll("kaerb", "break"); FileWriter writer = new FileWriter("archivo.cpp"); // leer archivo origen writer.write(newtext);writer.close(); } catch (IOException ioe) { ioe.printStackTrace(); } try { File file = new File("archivo.cpp"); // leer archivo origen </pre>
---	--

```

        BufferedReader reader
= new BufferedReader(new
        FileReader(file));

        String line14 = "",
        oldtext14 = "";

        while((line14 =
        reader.readLine()) != null)
        {
            oldtext14 += line14
            + "";
        }

        reader.close();

        String newtext =
        oldtext14.replaceAll("daer",
        "cin");

        FileWriter writer = new
        FileWriter("archivo.cpp"); //
        leer archivo origen

        writer.write(newtext);writer.clo
        se();
    }
    catch (IOException ioe)
    {

```

```

        ioe.printStackTrace();
    }
    try
    {
        File file = new
        File("archivo.cpp"); // leer
        archivo origen

        BufferedReader reader
        = new BufferedReader(new
        FileReader(file));

        String line15 = "",
        oldtext15 = "";

        while((line15 =
        reader.readLine()) != null)
        {
            oldtext15 += line15
            + "";
        }

        reader.close();

        String newtext =
        oldtext15.replaceAll("tni",
        "int");

```

<pre> FileWriter writer = new FileWriter("archivo.cpp"); // leer archivo origen writer.write(newtext);writer.close(); } catch (IOException ioe) { ioe.printStackTrace(); } try { File file = new File("archivo.cpp"); // escribir archivo origen BufferedReader reader = new BufferedReader(new FileReader(file)); String line16 = "", oldtext16 = ""; while((line16 = reader.readLine()) != null) { </pre>	<pre> oldtext16 += line16 + ""; } reader.close(); String newtext = oldtext16.replaceAll("rahc", "char"); FileWriter writer = new FileWriter("archivo.cpp"); // escribir archivo origen writer.write(newtext);writer.close(); } catch (IOException ioe) { ioe.printStackTrace(); } try { File file = new File("archivo.cpp"); // leer archivo origen </pre>
--	--

```

        BufferedReader reader
= new BufferedReader(new
    FileReader(file));

    String line17 = "",
oldtext17 = "";

    while((line17 =
reader.readLine()) != null)
        {
            oldtext17 += line17
+ "";
        }

    reader.close();

    String newtext =
oldtext17.replaceAll("ced",
"float");

    FileWriter writer = new
FileWriter("archivo.cpp"); //
escribir archivo origen

writer.write(newtext);writer.clo
se();
    }
catch (IOException ioe)
    {

```

```

        ioe.printStackTrace();
    }
    try
    {
        File file = new
File("archivo.cpp"); // leer
archivo origen

        BufferedReader reader
= new BufferedReader(new
FileReader(file));

        String line18 = "",
oldtext18 = "";

        while((line18 =
reader.readLine()) != null)
            {
                oldtext18 += line18
+ "";
            }

        reader.close();

        String newtext =
oldtext18.replaceAll("gnirts",
"string");

        FileWriter writer = new
FileWriter("archivo.cpp");

```

```

        writer.write(newtext);writer.close();
    }

    catch (IOException ioe)
    {
        ioe.printStackTrace();
    }

    try
    {
        File file = new
        File("archivo.cpp"); // leer
        archivo origen

        BufferedReader reader
        = new BufferedReader(new
        FileReader(file));

        String line19 = "",
        oldtext19 = "";

        while((line19 =
        reader.readLine()) != null)
        {
            oldtext19 += line19
            + "";
        }
    }

```

```

        reader.close();

        String newtext =
        oldtext19.replaceAll("amargor
        p", "/*Programa*/");

        FileWriter writer = new
        FileWriter("archivo.cpp");

        writer.write(newtext);writer.close();
    }

    catch (IOException ioe)
    {
        ioe.printStackTrace();
    }

    try
    {
        File file = new
        File("archivo.cpp"); // leer
        archivo origen

        BufferedReader reader
        = new BufferedReader(new
        FileReader(file));

        String line20 = "",
        oldtext20 = "";
    }

```

```

        while((line20 =
reader.readLine()) != null)
        {
            oldtext20 += line20
            + "";
        }

        reader.close();

        String newtext =
oldtext20.replaceAll("tluafed",
"default");

        FileWriter writer = new
        FileWriter("archivo.cpp");

        writer.write(newtext);writer.close();
    }
    catch (IOException ioe)
    {
        ioe.printStackTrace();
    }
}

```

Clase Crear2: Esta clase genera un archivo nuevo .cpp

```

package Traductor;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

//*****Genera nuevo archivo .cpp

public class crear2 {

    public static void main(String[] args) throws IOException {

```

```

        String ruta =
"/Users/pablomontes/Documents/Programas/konpaira/Final.cpp";

        File archivo = new File(ruta);

        BufferedWriter bw;

        if(archivo.exists()) {

bw = new BufferedWriter(new FileWriter(archivo));

        } else {

bw = new BufferedWriter(new FileWriter(archivo));

        bw.close();

        }

    }

```

**Clase Escribir: Esta clase genera
las librerías de nuestro archivo
.cpp**

```

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

```

コンパイラ

**public class escribir {
//***** Genero las librerías, da un salto
de linea y copea tal cual el archivo .cpp**

```
    public static void main(String[] args) throws IOException  
    {  
        File file = new File("archivo.cpp");  
        BufferedReader br = new BufferedReader(new FileReader(file));  
        String linea = br.readLine();  
        String todo = "";  
        try  
        {  
            while(linea !=null)  
            {  
                todo = todo + linea + "\n";  
                linea = br.readLine();  
            }  
            br.close();  
        }  
        catch (IOException e)  
        {  
        }  
        File TextFile = new File("Final.cpp");
```

```
FileWriter TextOut = new FileWriter(TextFile, true);  
    TextOut.write("#include<stdlib.h>\n");  
    TextOut.write("#include<stdio.h>\n");  
    TextOut.write("#include<string.h>\n");  
    TextOut.write("#include<iostream>\n");  
    TextOut.write("#include<cstdlib>\n");  
    TextOut.write("using namespace std;\n");  
        TextOut.write(todo);  
        TextOut.close();  
    }  
}
```

コンパイラ



Diccionario de Datos

Lenguaje c	Lenguaje kompaira
if	fi
while	elihw
else	esle
for	rof
printf	etihw
do	od
void	diov
public	cilbup
private	etavirp
static	citats
main	niam
switch	hctiws
case	esac
break	kaerb
cin	daer
int	tni
char	rahc
dec	ced
string	gnirts
programa	amargorp

Conclusiones a cerca del compilador:

MONTES SIERRA PABLO:

Lo que logramos como equipo, al realizar el compilador es darnos cuenta de lo que hay detrás de todo un entorno de desarrollo y que realmente la IDE no es lo importante. El compilador de cada IDE, es lo mas importante, generamos léxica, semántica y sintáctica, para nuestro propio lenguaje de programación, donde se tenia que comprobar las tres fases, para indicarnos que nuestro código ya escrito en nuestro lenguaje de programación no presentaba ningún error, en caso contrario hacer las correcciones debidas, ya no solo programamos, sabemos ahora todo lo que viene detrás de la programación, así como la generación de código intermedio, ensamblador, objeto y ya el resultado final que es el exe.

RODRIGUEZ APARICIO YAIR:

Como conclusión a cerca de nuestro compilador puedo decir que logramos terminarlo en tiempo y forma, cabe destacar que el código generado en este compilador fue hecho por nosotros 3, nadie fue por las tortas todos le dimos y eso fue lo mejor de todo el “trabajo en equipo y lograr tener un compilador”.

SANTAMARIA GUERRERO DIEGO:

En la materia de Compiladores se llevó a cabo el desarrollo de un compilador. Un compilador está compuesto de dos partes: el análisis y la síntesis. El análisis es se divide en tres partes; la léxica, la sintaxis y la semántica. La síntesis también se divide en tres partes; el código intermedio, el optimizador del código y el código objeto.

Con el desarrollo del compilador podemos concluir que se cumplió con el objetivo de la materia, ya que la construcción de un compilador requiere de conocimientos previos, como; la teoría de autómatas, estructuras de datos, programación orientada a objetos y lenguajes de bajo nivel.

Se aprendió a trabajar en equipo, a plantearse el problema antes de sentarse en una computadora a tratar de resolverlo. Aprendimos que para la solución de un problema, es necesario escribirlo o dibujarlo y pasarlo a un lenguaje de programación es solo la decodificación.