

PLAYER 1



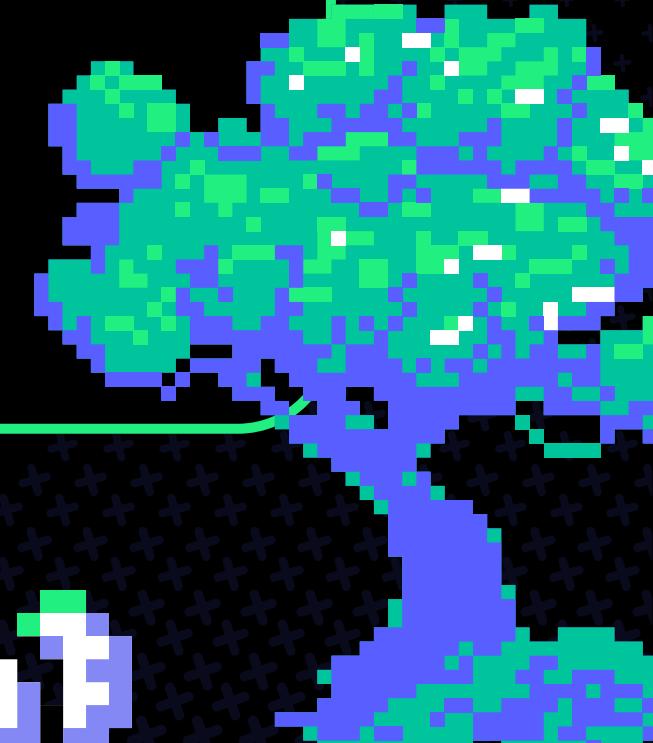
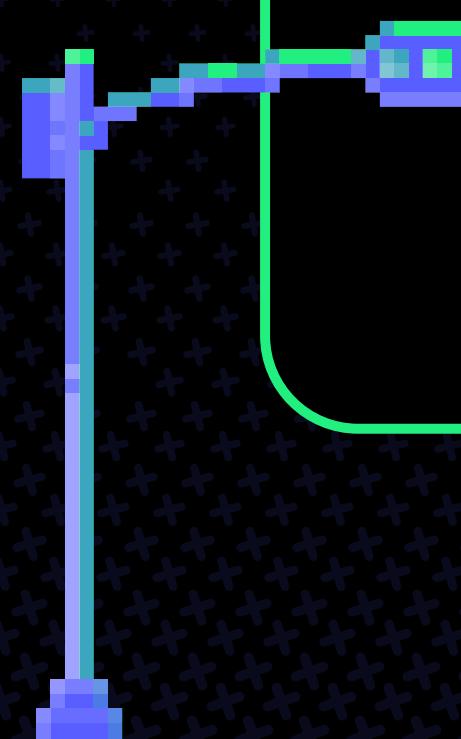
HIGHSCORE 2500



PLAYER 2

SESSION #1

◆ INTRODUCCIÓN A C++



01

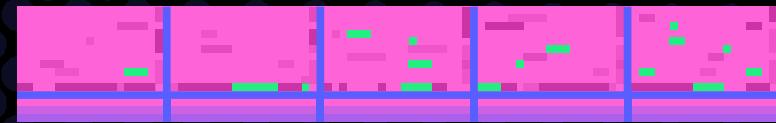
07

12

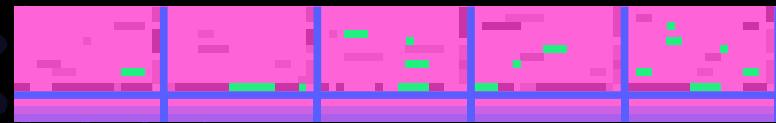


TEMAS

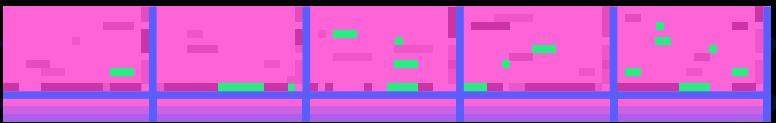
◆ ¿QUÉ ESTAREMOS VIENDO?



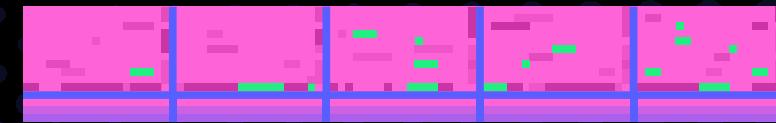
CONCEPTOS
BÁSICOS



TIPOS DE
VARIABLES



OPERACIONES



CONDICIONALES Y
LOOPS



EXTRA

SUIT DE JETBRAINS

1 Student Pack License

 **JetBrains Product Pack for Students**

[Download](#) ▾

Licensed to: Juan José López Pérez

License restriction: For educational use only

Valid through: February 05, 2024

Following products included:

• CLion	• DataGrip	• DataSpell	• dotCover	• dotMemory
• dotTrace	• GoLand	• IntelliJ IDEA Ultimate	• PhpStorm	• PyCharm
• ReSharper	• ReSharper C++	• Rider	• RubyMine	• WebStorm

After downloading and installing the software, simply run it and follow the on-screen prompts to sign in with your JetBrains Account.

[Buy new license](#)

Download CLion 2023.2.2

[WINDOWS](#) [MAC](#) [LINUX](#)

Product: [CLion](#)

Version: 2023.2.2

Build: 232.9921.42

Released: September 13, 2023

[DOWNLOAD](#)

714.58 MB

[SHA256 checksum](#)

[DOWNLOAD \(ARM64\)](#)

591.65 MB

[SHA256 checksum](#)

[DOWNLOAD .ZIP](#)

1028.59 MB

[SHA256 checksum](#)

CONCEPTOS BÁSICOS



NAMESPACES Y HEADERS

LOS NAMESPACES ES UNA REGIÓN DECLARATIVA QUE PROPORCIONA UN ALCANCE A LOS IDENTIFICADORES DENTRO DE ELLA.

```
USING NAMESPACE STD;  
  
NAMESPACE SALUDO  
{  
    VOID SALUDAR (){  
        COUT << "HOLA";  
    }  
}  
  
USING NAMESPACE SALUDO;  
  
INT MAIN (){  
    SALUDO::SALUDAR();  
    //SALIDA: HOLA  
    RETURN 0;  
}
```

#INCLUDE SE UTILIZA PARA IMPORTAR Y PODER UTILIZAR FUNCIONES Y VARIABLES DEFINIDAS EN OTROS ARCHIVOS

```
#INCLUDE <iostream>  
  
INT MAIN ()  
{  
    STD::COUT << "HELLO WORLD!";  
    RETURN 0;  
}
```

COMENTARIOS

SIEMPRE ES BUENO TENER REGISTRO POR MEDIO DE COMENTARIOS SOBRE LO QUE HACE NUESTRO CODIGO.

```
#INCLUDE <iostream>
```

```
/* EL ENCABEZADO <iostream> DEFINE LOS OBJETOS DE FLUJO ESTÁNDAR QUE INGRESAN Y EMITEN DATOS. */
```

```
USING NAMESPACE STD;
```

```
INT MAIN () {
    COUT << "HELLO WORLD!";
    // LA SALIDA ES "HELLO WORLD"
    RETURN 0;
}
```

```
// COMENTARIOS DE UNA LINEA
```

```
/* COMENTARIOS MULTILINEA */
```

DECLARACIÓN Y ASIGNACIÓN

C++ ES UN LENGUAJE TIPOADO

```
INT A = 10;  
  
A = A + 5;  
  
STD :: COUT << A;  
  
A = "HOLA";  
  
STD :: COUT << A;
```

JAVASCRIPT NO LO ES

```
A = 10  
  
A += 5  
  
CONSOLE.LOG(A)  
  
A = "HOLA" + 1  
  
CONSOLE.LOG(A)
```

TIPOSOE VARIABLES



TIPOS ALFANUMÉRICOS

STRING (CONJUNTO DE CARACTERES)

CHAR (UN SOLO CARACTER)

```
STRING HI = "HOLA";
```

```
CHAR B = 'B';
```

TIPOS NUMÉRICOS

INTEGERS (NUMEROS ENTEROS)

FLOATING (7 DECIMALES PRECISION)

DOUBLE (15 DECIMALES PRECISION)

INT A = 42;

FLOAT B = 5.99;

DOUBLE C = 6.4945;

TIPOS BOOLEANOS

```
BOOL V = TRUE;  
BOOL F = FALSE;
```

OPERACIONES



A
R
I
T
M

E
T
I
C
A

SUMA

```
INT A = 5;  
INT B = 6;  
INT C = A + B;
```

RESTA

```
INT A = 5;  
INT B = 6;  
INT C = B - A;
```

MULTI

```
INT A = 5;  
INT B = 6;  
INT C = A * B;
```

DIVISION

```
INT A = 10;  
INT B = 2;  
INT C = A / B;
```

A
R
I
T
M
E
T
I
C
A

MODULO

```
INT A = 10;  
INT B = 2;  
INT C = A % B;
```

AUMENTAR DISMINUIR

```
INT C = 2;  
++C;
```

```
INT C = 2;  
--C;
```

A
S
I
G
N

IGUALDAD

SUMA/RESTA

SIGNO =

```
INT A = 5;
```

SIGNO +=

```
INT A += 5;  
INT A = A + 5;
```

SIGNO -=

```
INT A -= 5;  
INT A = A - 5;
```

A
C
I
O
N

MODULO

MULTI/DIV

SIGNO %≡

```
INT A %= 5;  
INT A = A % 5;
```

SIGNO *=

```
INT A *= 5;  
INT A = A * 5;
```

SIGNO /=

```
INT A /= 5;  
INT A = A / 5;
```

C
O
M
P
A
R
A
C
I
O
N

IGUAL A / MAYOR QUE / MENOR QUE

SIGNO ==

```
INT A = 4 == 5;
```

SIGNO >

```
INT A = 5 > 2;
```

SIGNO <

```
INT A = 5 < 2;
```

NO ES IGUAL

SIGNO !=

```
INT A = 4 != 5;
```

MAYOR O IGUAL QUE / MENOR O IGUAL QUE

SIGNO >=

```
INT A = 5 >= 2; INT A = 5 <= 2;
```

SIGNO <=

L
O
G
I
C
A
S

AND

```
INT A = 10;  
INT B = 2;  
INT C = A < 5 && B < 10;
```

NOT

```
INT A = 10;  
INT B = 2;  
INT C = !(A < 5 && B < 10);
```

OR

```
INT A = 10;  
INT B = 2;  
INT C = A < 5 || B < 10;
```

CONDICIONALES Y BUCLE



IF

UTILIZADO CUANDO LA
CONDICIÓN ES VERDADERA

```
IF ( CONDICION ) {  
    // BLOQUE DE CÓDIGO A EJECUTAR  
}
```

ELSE

UTILIZADO CUANDO LA
CONDICIÓN ES FALSA

```
IF ( CONDICION ) {  
    // BLOQUE DE CÓDIGO A EJECUTAR  
} ELSE {  
    // BLOQUE DE CÓDIGO A EJECUTAR  
}
```

ELSE IF

UTILIZADO CUANDO SE REQUIERE DE OTRA CONDICIÓN

OPERADOR TERNARIO

FORMA ABREVIADA DE UN IF Y UN ELSE

```
IF ( CONDICION 1 ) {
    // BLOQUE DE CÓDIGO A EJECUTAR
} ELSE IF ( CONDICIÓN 2 ) {
    // BLOQUE DE CÓDIGO A EJECUTAR
} ELSE {
    // BLOQUE DE CÓDIGO A EJECUTAR
}
```

```
TYPE VARIABLE = (CONDITION) ?  
EXPRESIONTRUE : EXPRESIONFALSE;
```

```
INT HORA = 16;
```

```
STRING RESULT = (HORA < 12) ?  
"BUENOS DIAS." : "BUENAS TARDES.";
```

SWITCH

UTILIZADO CUANDO SE REQUIERE DE OTRA CONDICIÓN

```
SWITCH (EXPRESION) {  
    CASE X:  
        // CODIGO  
        BREAK;  
    CASE Y:  
        // CODIGO  
        BREAK;  
    DEFAULT:  
        // CODIGO  
}
```

WHILE

LOOP HASTA QUE SE CUMPLE UNA CONDICIÓN

```
WHILE (CONDICION) {  
    // CODIGO  
}
```

FOR

LOOP REPETIDO N VECES,
DETERMINADO POR UN
INICIO Y UN SALTO.

```
FOR (INICIO ; CONDICION; SALTO) {  
    // CODIGO  
}
```

DO WHILE

LOOP HASTA QUE SE
CUMPLE UNA CONDICIÓN,
EJECUTANDO ANTES DE
REVISAR LA CONDICIÓN

```
DO {  
    // CODIGO  
} WHILE (CONDICION) ;
```

CONTINUE

CUANDO EN ALGUN PUNTO DE NUESTRO BUCLE NO
NECESITAMOS QUE HAGA ALGO NUESTRO CODIGO,
UTILIZAMOS ESTA SENTENCIA DE CONTROL PARA
SALTAR ESE CICLO.

CONTINUE;

BREAK

CUANDO LLEGAMOS A UN PUNTO QUE NOS
INTERESA EN EL BUCLE, PODEMOS UTILIZAR
ESTA SENTENCIA DE CONTROL PARA SALIR DE
BUCLE.

BREAK;

DUUDAS O COMENTARIOS?