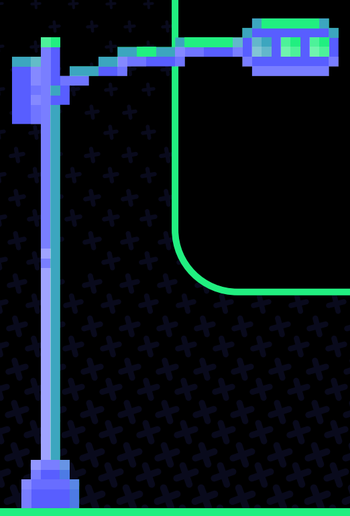
PLAYER 1 HIGHSCORE 2500 PLAYER 2

SESIÓN #2

- C++ APLICADO



01 07 12

ANUNCIOS

01 07 12

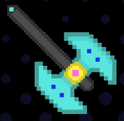
TEMAS

¿QUÉ ESTAREMOS VIENDO?

ARRAYS Y VECTORES

STRUCTS

REFERENCIAS



FUNCIONES

ARRAYS Y VECTORES

DECLARACIÓN Y ACCESO

```
STRING FRUTAS [ 4 ];

STRING MARCAS [ ];

STRING FRUTAS [ 4 ] = {"BANANO",
"PERA", "MANZANAS", "NARANJA"};

INT NUMEROS [ 3 ] = {10, 20, 30};
```

SE PUEDE DECLARAR UN ARRAY CON ITEMS, SIN ESPECIFICAR LA CANTIDAD EN LA VARIABLE O RESERVAR EL ESPACIO PARA POSTERIORMENTE LLENARLO

EL ACCESO ES POR MEDIO DE SUS POSICIONES CONTANDO DESDE CERO

PARA ACCEDER AL ITEM MANZANAS EN EL ARRAY DE FRUTAS ES LA POSICION 2

```
STD :: COUT << FRUTAS [ 2 ];

POS 0 --- BANANO
POS 1 --- PERA
POS 2 --- MANZANAS
POS 3 --- NARANJA
```

VECTORES Y ARRAYS COMO LIBRERIAS

```
#INCLUDE <VECTOR>
#INCLUDE <ARRAY>
```

```
#INCLUDE <VECTOR>

STD::VECTOR<INT> VEC;

VEC.PUSH_BACK(42);

VEC.POP_BACK();

INT ELEMENT = VEC.AT(0);

INT SIZE = VEC.SIZE();

BOOL ISEMPTY = VEC.EMPTY();

VEC.CLEAR();
```

```
#INCLUDE <ARRAY>

STD::ARRAY<INT, 5> ARR = {1, 2, 3, 4, 5};

INT VALOR = ARR.AT(2);

INT TAM = ARR.SIZE();

INT FRONT = ARR.FRONT();

INT BACK  = ARR.BACK();
```


STRUCTS

CREACIÓN

UN STRUCT ES UNA FORMA DE AGRUPAR ATRIBUTOS O CARACTERÍSTICAS.

```
STRUCT {
  INT MYNUM;
  STRING MYSTRING;
} MYSTRUCTURE;
```

ACCESO

TOMANDO LA LÓGICA DE LA PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

```
MYSTRUCTURE.MYNUM = 1;
MYSTRUCTURE.MYSTRING = "HELLO WORLD!";
```

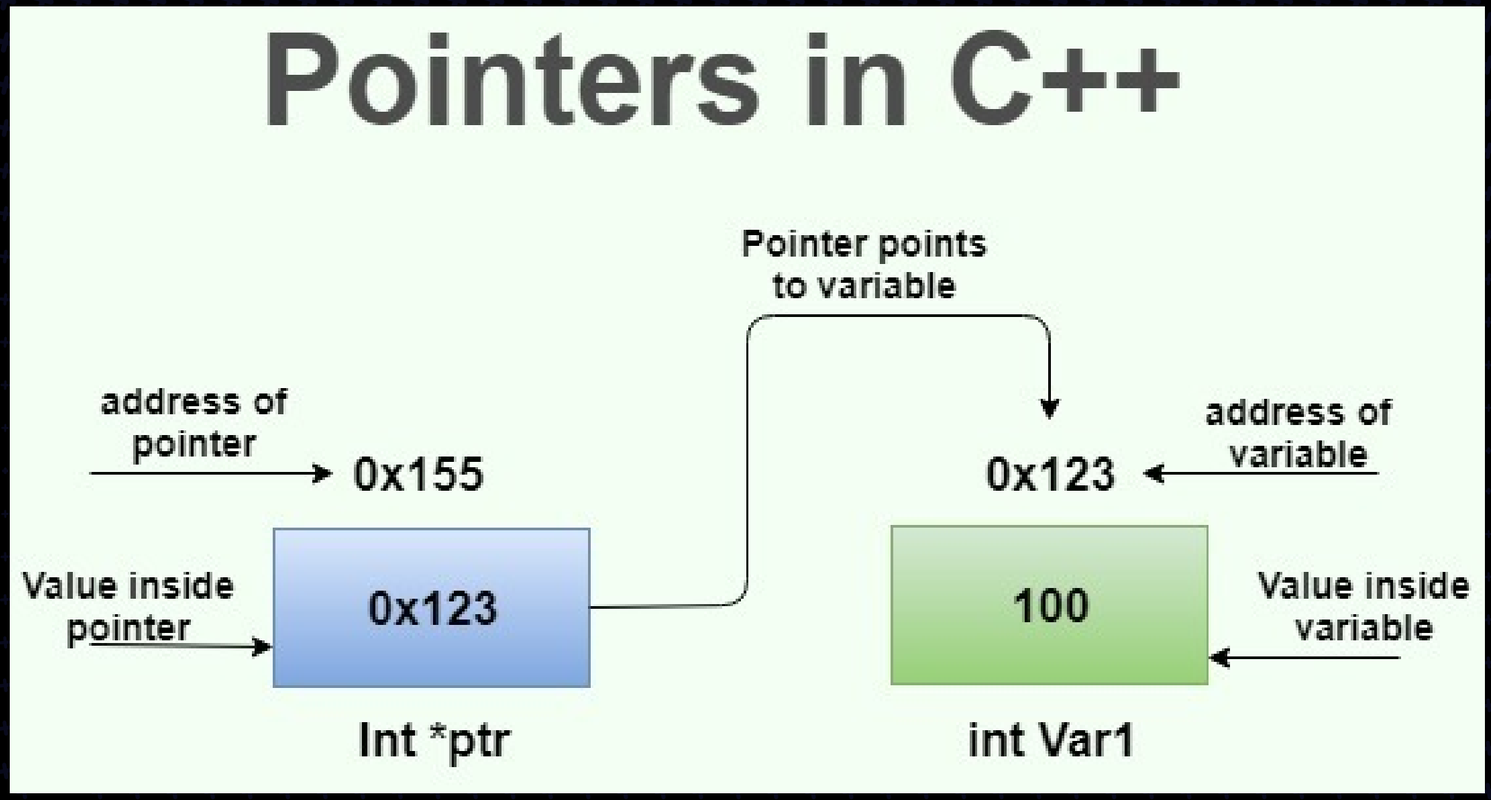
```
struct car {
  string brand;
  string model;
  int year;
};
```

INSTANCIA

```
car myCar1;
myCar1.brand = "BMW";
myCar1.model = "X5";
myCar1.year = 1999;
```

```
car myCar2;
myCar2.brand = "Ford";
myCar2.model = "Mustang";
myCar2.year = 1969;
```

REFERENCIAS


Pointers in C++
Pointer points to variable
address of pointer
0x155
address of variable
0x123
Value inside pointer
0x123
100
Value inside variable
Int *ptr
int Var1

EJEMPLO

```
INT A = 42;
INT *PTR;
PTR = &A;
INT NUM = *PTR;
```

VALOR = 42 DIRECCION = 0XD45F

VALOR = ? DIRECCION = ?

PTR APUNTA A LA DIRECCION DE A

NUM CONTIENE EL VALOR DE A

FUNCIONES

CREACION

SIN PARAMETROS Y SIN VALOR DE RETORNO

```
VOID MVFUNCTION ( ) {
  // CODIGO
}
```

CON PARAMETROS Y SIN VALOR DE RETORNO

```
VOID MVFUNCTION ( PARAM1, PARAM2 )
{
  // CODIGO
}
```

LO ÚNICO QUE CAMBIA EN EL CUERPO DE LA FUNCION CUANDO SI SE TIENE UN VALOR DE RETORNO ES COLOCAR LA PALABRA RESERVADA RETURN CON EL VALOR DE RESPUESTA

```
INT MVFUNCTION ( PARAM1, PARAM2 ) {
  // CODIGO
  RETURN RESPUESTA
}
```

LLAMADAS

CON Y SIN PARAMETROS SIN VALOR DE RETORNO

```
MYFUNCTION();
MYFUNCTION(PARAM1, PARAM2);
```

CON Y SIN PARAMETROS CON VALOR DE RETORNO

```
TYPE VAR = MYFUNCTION();
TYPE VAR = MYFUNCTION(PARAM1, PARAM2);
```

RECURSIVIDAD

ES LA CONSTANTE LLAMADA DE UNA FUNCIÓN DENTRO DE SI MISMA HASTA QUE ESTA CUMPLE SU OBJETIVO.

```
INT RESULT = SUM ( 10 );
COUT << RESULT;
```

¿DE QUÉ TRATA NUESTRA FUNCIÓN SUM?

```
INT SUM ( INT K ) {
    IF (K > 0) {
        RETURN K + SUM(K - 1);
    } ELSE {
        RETURN 0;
    }
}
```

EXPLICACION

SUM (10);
K > 0 ? RETURN K + SUM(K -1) : RETURN 0

10 + SUM(9)

10 + (9 + SUM(8))

10 + (9 + (8 + SUM(7)))

[SE REPITE HASTA QUE LA K <= 0]

10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + SUM(0)

10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0

SALIDA: 55

¿DUDAS O COMENTARIOS?