



Programación avanzada en C

Estructuras

Objetivos

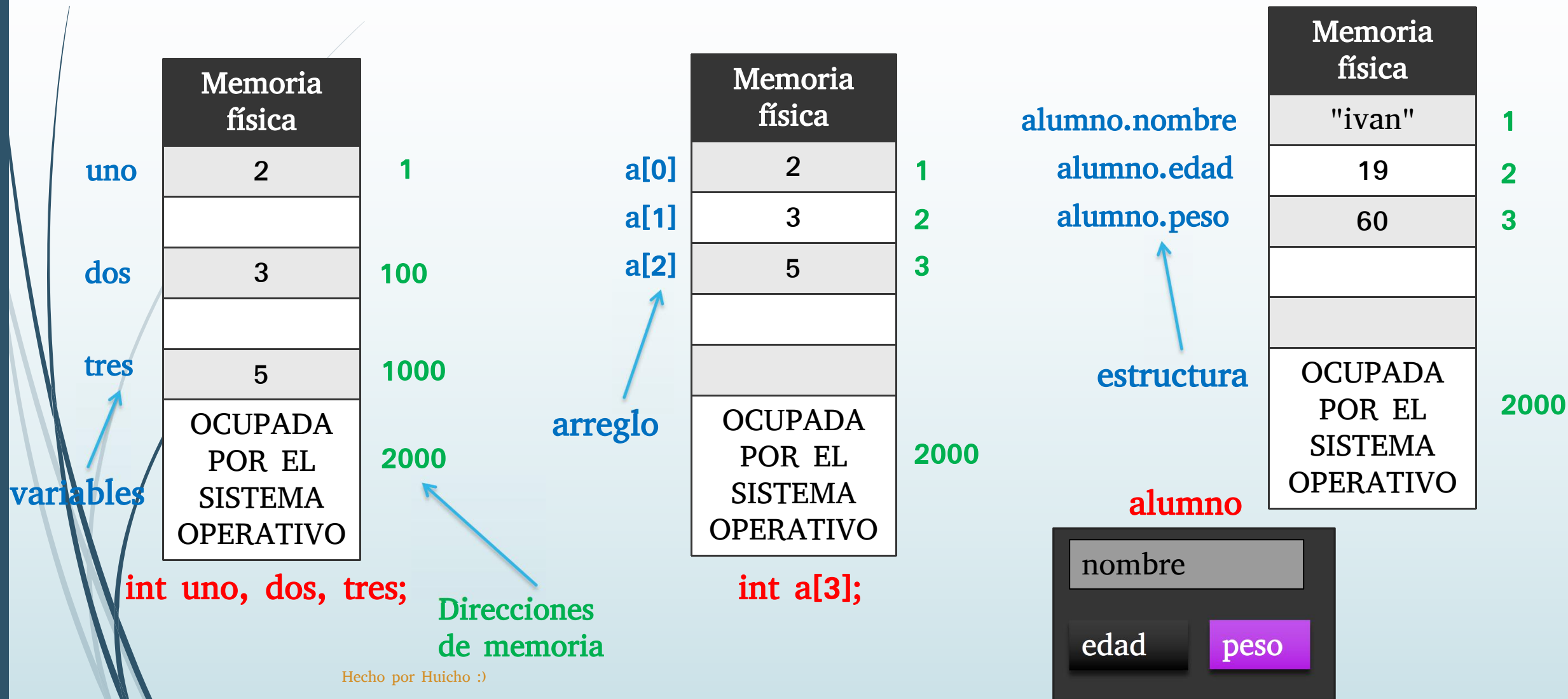
El alumno conocerá y aplicará los conceptos de arreglo y estructura en la realización de programas que resuelvan problemas de tipo numérico.

Recordar manejo de conceptos básicos de programación en lenguaje C.

Aprender el concepto de una estructura, sus variantes y su aplicación junto con otros elementos de programación.

Tipo de variable	Espacios de memoria	Tipos de datos	Ejemplo	Acceso
Variable mortal	1	1	<code>int var= 5;</code>	Por el nombre de la variable
Arreglo	Los solicitados en la declaración	1	<code>int arreglo[2]= {1,2};</code>	Por el nombre de la variable y posición del elemento
Estructura	Los solicitados en la declaración <small>Hecho por Huicho :)</small>	Los solicitados en la declaración	<code>struct { char nombre[50]; int edad; }alumno;</code>	Por el nombre de la variable "." y el nombre del subcuadrado

Variables VS Arreglo VS Estructura



Estructura

Tipo de dato nuevo diseñado y creado por el programador.

Podemos verla como una caja con compartimentos o cuadritos de diferentes tamaños para guardar cosas.

Para tener acceso a un cuadrito en específico se indica con el operador punto:

nombrecaja.cuadrato

```
printf("%d", alumno.edad);
```

```
alumno.edad= 29;
```

Declaración de estructura

Declarar dentro de la función como una variable normal indicando el nombre de la caja y sus compartimentos agrupados en llaves:

```
struct  
{  
    char nombre[50];  
    int edad;  
}alumno;
```

```
struct alumno  
{  
    char nombre[50];  
    int edad;  
};
```

Typedef

Es una forma de darle apodos a tipos de datos que ya están definidos.

Se usan normalmente en estructuras para omitir la palabra struct al crear una variable o arreglo de esa estructura.

```
struct alumno  
{  
    char nombre[50];  
    int edad;  
};
```

```
struct alumno a1;
```

```
typedef struct  
{  
    char nombre[50];  
    int edad;  
}alumno;
```

```
alumno a1;
```

Variables de estructura

Una vez declarada la plantilla de estructura se pueden crear ejemplares con la misma información pero con un nombre diferente:

```
struct
{
    char nombre[50];
    int edad;
}alumno;

struct alumno a1= {"Luis", 29};
```

Hecho por Huicho :)

```
struct alumno
{
    char nombre[50];
    int edad;
}a1;

a1.edad= 29;
```


Variables de estructura

```
#include <stdio.h>

int main(int argc, char*argv[])
{
    struct alumno
    {
        char nombre[50];
        int edad;
    };

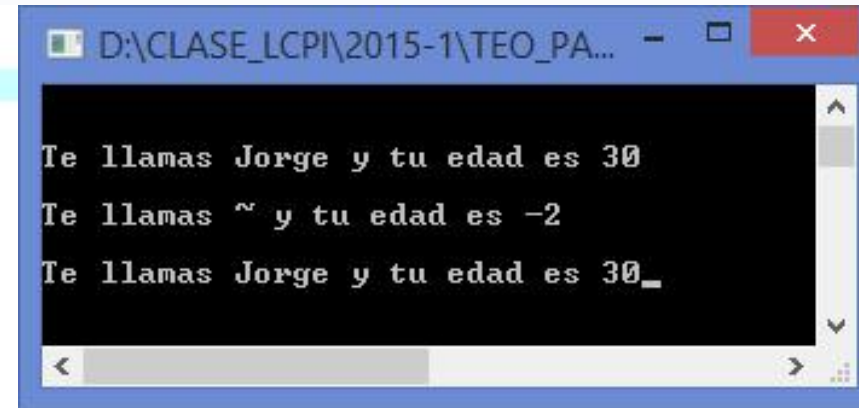
    struct alumno chavo1={"Jorge", 30};
    struct alumno chavo2; //estructura vacia

    printf("\n\nTe llamas %s y tu edad es %d", chavo1.nombre, chavo1.edad);

    //la estructura chavo2 imprime basura xq no esta inicializada ni se ha leido
    printf("\n\nTe llamas %s y tu edad es %d", chavo2.nombre, chavo2.edad);

    //pasando datos de chavo1 a chavo2
    chavo2=chavo1;
    printf("\n\nTe llamas %s y tu edad es %d", chavo2.nombre, chavo2.edad);

    getchar();
    return 0;
}
```



```
D:\CLASE_LCPI\2015-1\TEO_PA...
Te llamas Jorge y tu edad es 30
Te llamas ~ y tu edad es -2
Te llamas Jorge y tu edad es 30_
```

Coordenadas de una recta con estructuras

OPCIÓN 1

Declarar en una estructura los compartimentos para x1, x2, y1, y2

```
struct
{
    int x1;
    int x2;
    int y1;
    int y2;
}recta;
```

OPCIÓN 2

Declarar 2 variables de una estructura con los compartimentos x, y

```
struct punto
{
    int x;
    int y;
};

struct punto p1, p2;
```

Estructuras anidadas

Dentro de una estructura se puede declarar una variable de otra estructura ya declarada.

Para acceder a un compartimento de una estructura interna hay que hacerlo en orden, desde la más externa hasta la más interna.

```
struct persona
{
    char nombre[25];
    int edad;
    char estado; /* C = casado, S = soltero */
};
```

```
struct alumno
{
    struct persona p1;
    int cuenta;
    char carrera[25];
};
```

```
struct profe
{
    struct persona p2;
    int trabajador;
    char asignatura[25];
};
```

Estructuras anidadas

Inicializar los valores de la estructura anidada:

```
struct alumno alu1={{ "Ivan", 19, 's'}, 4040, "compu"};
```

Asignar individualmente valores a compartimentos anidados:

```
alu1.p1.edad= 19;
```

```
alu1.p1.estado='s';
```

En caso de asignación a cadenas usar función de copia de la librería string.h y NO directamente:

```
strcpy(alu1.p1.nombre, "Ivan");
```

Arreglos de estructuras

Teniendo una plantilla de estructura podemos crear variables de ella que lleven la misma definición pero si son muchas el manejo de los nombres se complica y mejor usamos arreglos:

```
struct alumno  
{  
    char nombre[50];  
    int edad;  
}a1, a2, a3;
```

```
a1.edad= 29;  
a2.edad= 18;  
a3.edad= 22;
```

Hecho por Huicho :)

```
struct alumno  
{  
    char nombre[50];  
    int edad;  
}alu[3];
```

```
alu[0].edad= 29;  
alu[1].edad= 18;  
alu[2].edad= 22;
```

```
struct alumno alu[3];
```

Funciones y estructuras

```
#include<stdio.h>

void pasaEstruct(int e)
{
    printf("\n\n\n\t La edad pasada desde funcion es: %d", e);
}

int main(int argc, char*argv[])
{
    struct
    {
        char nombre[50];
        int edad;
    }alumno;

    printf("Este programa guarda datos de alumno en una estructura \n\n");

    printf("\nDame tu nombre: ");
    scanf(" %[^\\n]",alumno.nombre);

    printf("\nDame tu edad: ");
    scanf("%d", &alumno.edad);

    printf("\n\nTe llamas %s y tu edad es %d", alumno.nombre, alumno.edad);

    pasaEstruct(alumno.edad);

    getchar();
    getchar();
    return 0;
}
```

Hecho por Huicho :)



```
/bin/bash
Este programa guarda datos de alumno en una estructura

Dame tu nombre: Jorge Luis
Dame tu edad: 33

Te llamas Jorge Luis y tu edad es 33

La edad pasada desde funcion es: 33
```

Funciones y estructuras

```
#include<stdio.h>

struct alumno
{
    char nombre[50];
    int edad;
};

struct alumno leer_estructura(void)
{
    struct alumno alu;

    printf("\nDame tu nombre: ");
    scanf(" %[^\\n]", alu.nombre);
    printf("\nDame tu edad: ");
    scanf("%d", &alu.edad);

    return alu;
}

void ver_estructura(struct alumno al)
{
    printf("\n\nTe llamas %s y tu edad es %d", al.nombre, al.edad);
}

int main(int argc, char*argv[])
{
    struct alumno a;

    a= leer_estructura();
    ver_estructura(a);

    getchar();
    getchar();
    return 0;
}
```

A terminal window titled "/bin/bash" with a red title bar. It shows the execution of the program. The user is prompted for their name and age. The output shows the name "Jorge Luis" and age "33". The final output is "Te llamas Jorge Luis y tu edad es 33".

```
/bin/bash

Dame tu nombre: Jorge Luis
Dame tu edad: 33

Te llamas Jorge Luis y tu edad es 33
```