

Programación Avanzada (TC2025)

Tema 1. Programación en lenguaje C

Tec de Monterrey. Campus Santa Fe
Departamento de Computación
Dr. Vicente Cubells (vcubells@tec.mx)

Temario

- Introducción a la materia
- Introducción al lenguaje C
- Evolución del lenguaje
- Diferencias con Java y C#
- Tipos de datos
- Operadores
- Algunas estructuras de control
- Introducción a las funciones
- Sintaxis
- Edición, compilación y enlace
- Estructura de un programa en C
- Ejemplos

Introducción a TC2025

Tema 1. Programación en lenguaje C

Tema 2. Arquitectura de un sistema operativo

Tema 3. Administración de procesos

Tema 4. Eventos y señales

Tema 5. Programación concurrente

Tema 6. Programación paralela

Proyecto final...

- Se desarrollará en equipos de dos (2) personas.
- La codificación se realizará en C, utilizando OpenMP y MPI.
- El proyecto debe funcionar en un clúster de cualquier tamaño compuesto por al menos 2 PCs.
- La entrega final incluye una presentación del proyecto al terminar el curso.
- Se debe subir el código fuente, la documentación y todo lo relacionado con el proyecto a GitHub.

Proyecto final

Concepto	%
Originalidad de la solución	5%
Complejidad del software	20%
Funcionalidad y operatividad del software	50%
Calidad del software	10%
Presentación y exposición	15%
Proyecto final	100%

Ejemplo de un programa en C

```
#include <stdio.h>
```

```
/* la función main inicia la ejecución del programa */
```

```
int main()
```

```
{
```

```
    int entero1; /* primer número introducido por el usuario */
```

```
    int entero2; /* segundo número introducido por el usuario */
```

```
    int suma;    /* variable en la cual se almacena la suma */
```

```
    printf( "Introduzca el primer entero\n" ); /* indicador */
```

```
    scanf( "%d", &entero1 ); /* lee un entero */
```

```
    printf( "Introduzca el segundo entero\n" ); /* indicador */
```

```
    scanf( "%d", &entero2 ); /* lee un entero */
```

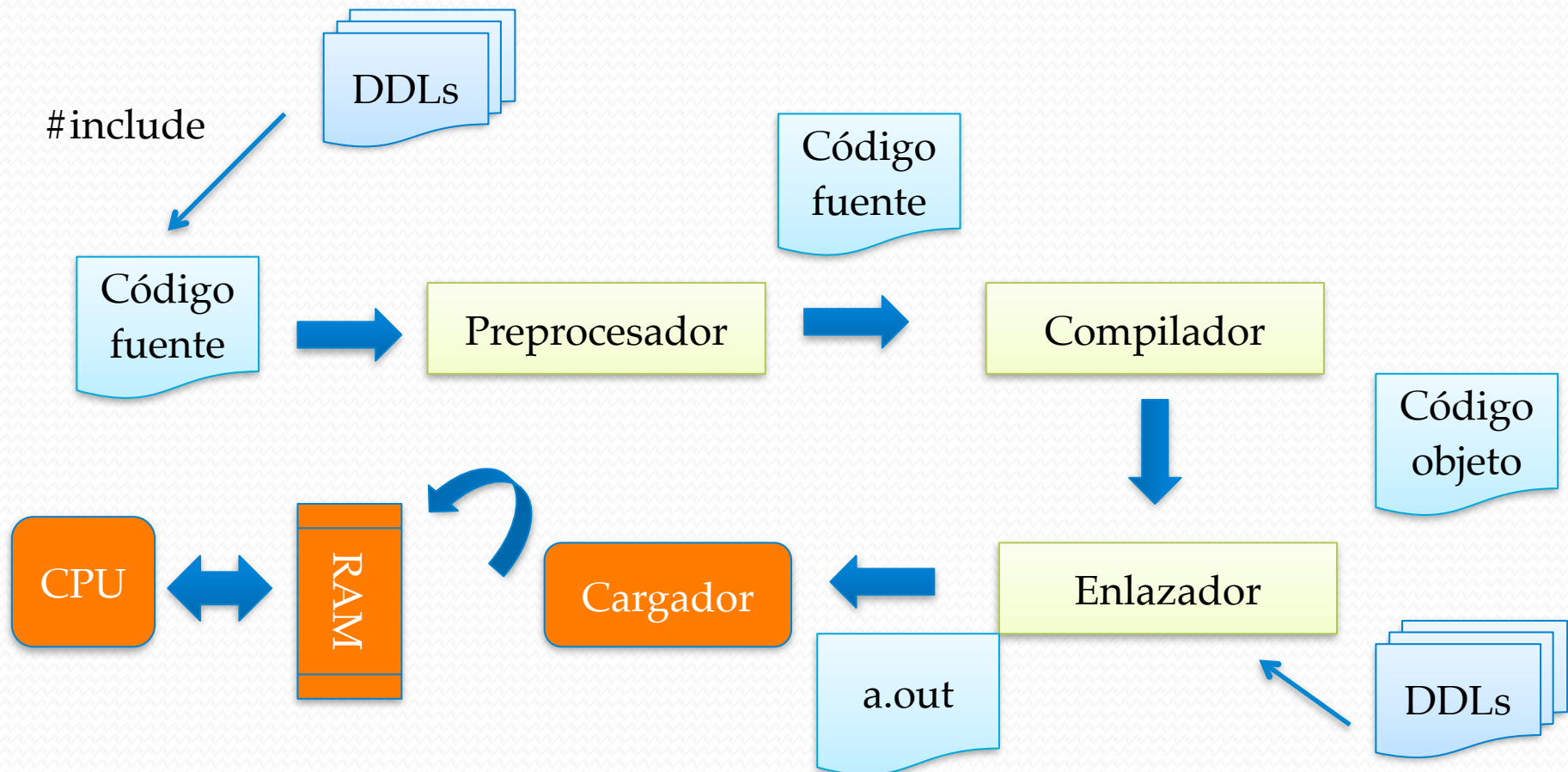
```
    suma = entero1 + entero2; /* asigna el total a suma */
```

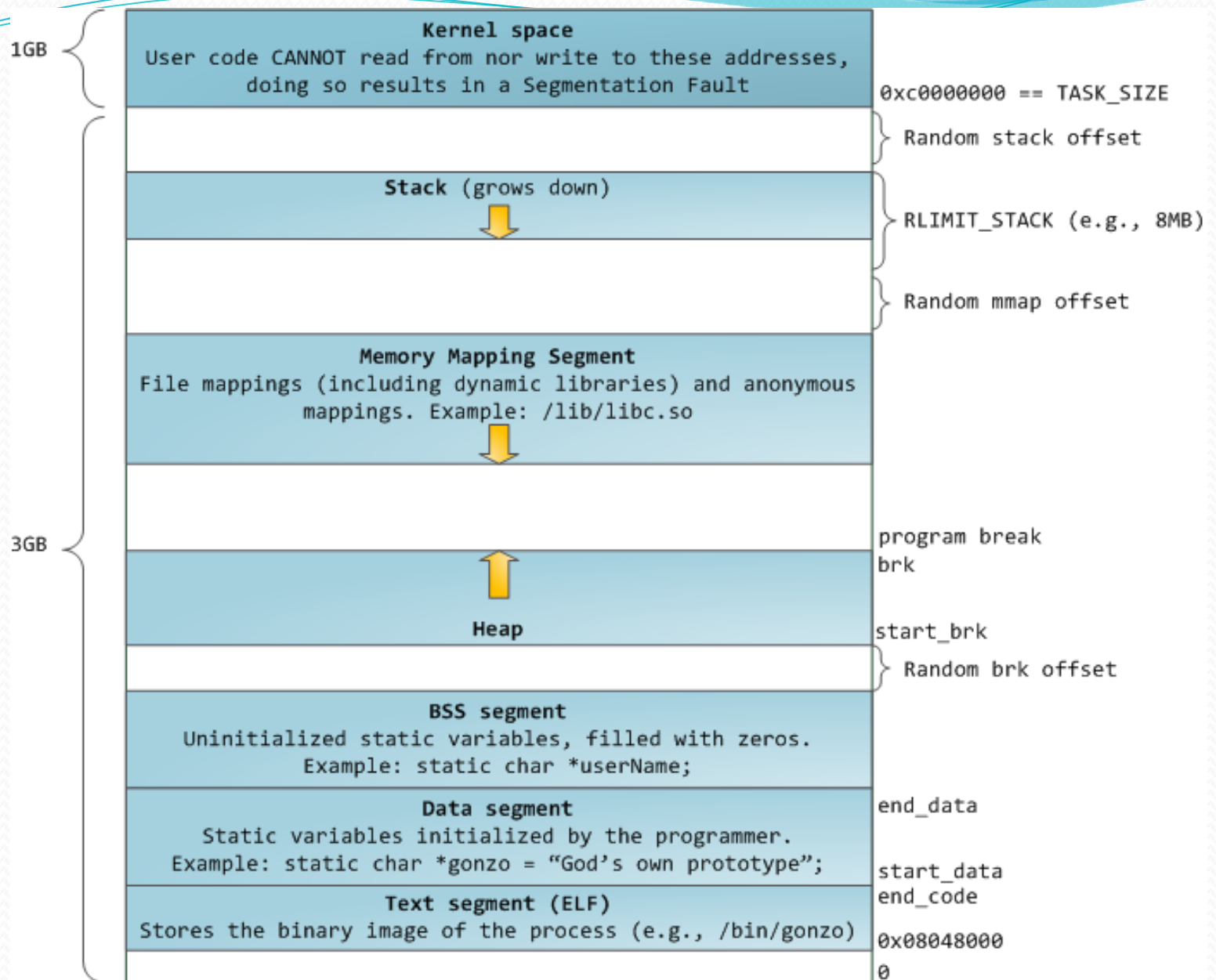
```
    printf( "La suma es %d\n", suma ); /* imprime la suma */
```

```
    return 0; /* indica que el programa terminó con éxito */
```

```
} /* fin de la función main */
```

Edición, compilación y enlace





Tres tipos de almacenamiento...

- Almacenamiento automático (stack)
 - Ejemplos
 - Variables locales
 - Argumentos de las funciones
 - Se crean automáticamente al inicio del bloque
 - Se destruyen automáticamente al finalizar el bloque
 - Su valor predeterminado es indeterminado

Tres tipos de almacenamiento...

- Almacenamiento estático
 - Ejemplos
 - Variables estáticas
 - La dirección de un objeto estático es la misma durante la ejecución del programa
 - Son inicializados a ceros binarios

Tres tipos de almacenamiento...

- Ejemplo de Almacenamiento estático

```
int num;
```

```
int func() {
```

```
    static int calls;
```

```
    return calls++;
```

```
}
```

// Se incializa en 0

Tres tipos de almacenamiento

- Almacenamiento dinámico (heap)
 - Ejemplos:
 - variables creadas con la función `malloc`
 - Persisten en memoria hasta que sean liberadas explícitamente mediante la función `free`
 - La memoria ocupada por este tipo de variables no es liberada al SO automáticamente al terminar la ejecución del programa
 - Su dirección de memoria se determina en tiempo de ejecución

Tipos de datos compuestos

- Estructuras
- Uniones
- Enumeraciones

Estructuras...

- Agrupación de datos de tipos diferentes
- Registros que agrupan datos primitivos para modelar un objeto complejo en forma de registro

```
struct libro
{
    int paginas;
    char titulo[25];
    char autor[30];
} programacion;
```

```
struct libro
{
    int paginas;
    char titulo[25];
    char autor[30];
};

struct libro programación;
```

Estructuras...

- Agrupación de datos de tipos diferentes
- Registros que agrupan datos primitivos para modelar un objeto complejo en forma de registro

```
typedef struct  
{  
    int paginas;  
    char titulo[25];  
    char autor[30];  
} libro;  
  
libro programacion;
```

Estructuras...

- Ejemplo de acceso a los miembros

```
#include <stdio.h>
#include <string.h>

typedef struct
{
    int paginas;
    char titulo[25];
    char autor[30];
} libro;

libro programacion;

int main()
{
    programacion.paginas = 350;
    strcpy(programacion.titulo, "Programación Avanzada");

    return 0;
}
```


Estructuras...

- Estructuras anidadas

```
struct autor
{
    int edad;
    char nombre[25];
    char apellidos[30];
};
```

```
struct libro
{
    int paginas;
    char titulo[25];
    struct autor escritor;
};
```

```
struct libro programación;
```

Estructuras

- Estructuras anidadas

```
#include <stdio.h>
#include <string.h>

/* Aquí van las declaraciones de las estructuras anteriores */

int main()
{
    programacion.paginas = 350;
    strcpy(programacion.titulo, "Programación Avanzada");
    strcpy(programacion.escriptor.nombre, "Deitel");

    printf("El autor es: %s \n", programacion.escriptor.nombre);

    return 0;
}
```

Uniones...

- Similar a las estructuras
- En una estructura cada miembro ocupa un área de memoria diferente
- En una unión todos los miembros ocupan el mismo espacio de memoria
 - Modificar uno, significa modificar el otro
 - Sólo se puede acceder a un miembro a la vez
 - Ocupa el espacio del miembro de mayor tamaño

Uniones

- Ejemplo

```
#include <stdio.h>

union libro
{
    int paginas;
    char titulo[25];
    char autor[30];
} programacion;

int main()
{
    programacion.paginas = 350;
    strcpy(programacion.titulo, "Programación Avanzada");
    strcpy(programacion.autor, "Deitel");

    printf("El título es: %s \n", programacion.titulo);

    return 0;
}
```



Se imprime
"Deitel"

Ejemplo

- Programar una aplicación que muestre la diferencia entre estructuras y uniones

Enumeraciones...

- Tipo de dato definido con constantes enteras

```
enum Nombre
{
    enumerador1 = valor_constantel,
    enumerador2 = valor_constante2,
    ...
    enumeradorn = valor_constanten,
};
```

Enumeraciones...

- Ejemplos

```
enum Boolean
{
    FALSE,
    TRUE
};

int main()
{
    enum Boolean existe = FALSE;
    while (existe != TRUE )
    {
        /* Hacer algo */
    }
    return 0;
}
```

Enumeraciones

- Ejemplos

```
enum DiasSemanas
{
    Domingo = 2,
    Lunes,
    Marte,
    Miercoles,
    Jueves,
    Viernes,
    Sabado
};

int main(int argc, char** argv)
{
    enum DiasSemanas dia;

    for (dia = Domingo; dia <= Sabado; dia++)
    {
        printf("%d ", dia);
    }

    return 0;
}
```


Resumiendo

- Las estructuras, uniones y enumeraciones son tipos de datos que nos permiten agrupar tipos primitivos en forma de registros
- En las estructuras cada miembro tiene su propio espacio de memoria
- En las uniones todos los miembros comparten el mismo espacio de memoria
- Las enumeraciones son agrupaciones de constantes enteras, que permiten asociar un valor numérico a un identificador de texto

Ejercicio

- Realice un programa que permita entrar un listado de personas (de cada persona se conoce su nombre, apellidos y edad) y permita:
 - Determinar la edad promedio
 - La persona más joven
 - La persona más vieja
 - Todas las personas que se encuentran en un rango de edades