# Función scanf

# Función scanf

También se pueden ingresar más valores por sentencia

# Función scanf

También se pueden ingresar más valores por sentencia

```
scanf("%d %d", &var1, &var2);
```

# Función scanf

También se pueden ingresar más valores por sentencia

```
scanf("%d %d", &var1, &var2);
```

donde para diferenciar los valores desde el teclado se ingresan con un espacio, un tab o un enter entre ellos.

# Función putchar

# Función putchar

```c
#include <stdio.h>
// u3-putchar.c

int main (void)
{
  int numero;

  printf("Ingrese un número (1-127): ");
  scanf("%d", &numero);

  printf("En la tabla ASCII: ");
  putchar(numero);

  return 0;
}
```

# Función putchar

```c
#include <stdio.h>
// u3-putchar.c

int main (void)
{
  int numero;

  printf("Ingrese un número (1-127): ");
  scanf("%d", &numero);

  printf("En la tabla ASCII: ");
  putchar(numero);

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u3-putchar.c
$
```

# Función putchar

```c
#include <stdio.h>
// u3-putchar.c

int main (void)
{
  int numero;

  printf("Ingrese un número (1-127): ");
  scanf("%d", &numero);

  printf("En la tabla ASCII: ");
  putchar(numero);

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u3-putchar.c
$ ./a.out
```

# Función putchar

```c
#include <stdio.h>
// u3-putchar.c

int main (void)
{
  int numero;

  printf("Ingrese un número (1-127): ");
  scanf("%d", &numero);

  printf("En la tabla ASCII: ");
  putchar(numero);

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u3-putchar.c
$ ./a.out
Ingrese un número (1-127):
```

# Función putchar

```c
#include <stdio.h>
// u3-putchar.c

int main (void)
{
  int numero;

  printf("Ingrese un número (1-127): ");
  scanf("%d", &numero);

  printf("En la tabla ASCII: ");
  putchar(numero);

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u3-putchar.c
$ ./a.out
Ingrese un número (1-127): 65
```

# Función putchar

```c
#include <stdio.h>
// u3-putchar.c

int main (void)
{
  int numero;

  printf("Ingrese un número (1-127): ");
  scanf("%d", &numero);

  printf("En la tabla ASCII: ");
  putchar(numero);

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u3-putchar.c
$ ./a.out
Ingrese un número (1-127): 65
En la tabla ASCII: A
$
```

# Función putchar

# Función putchar

La función `putchar` tiene el mismo efecto que `printf` si solo se imprime un caracter con `"%c"`

# Función putchar

La función `putchar` tiene el mismo efecto que `printf` si solo se imprime un caracter con `"%c"`

La sentencia

# Función putchar

La función `putchar` tiene el mismo efecto que `printf` si solo se imprime un caracter con "%c"

La sentencia

```
putchar(65);
```

# Función putchar

La función `putchar` tiene el mismo efecto que `printf` si solo se imprime un caracter con "%c"

La sentencia

```
putchar(65);
```

Tiene el mismo efecto que la sentencia

# Función putchar

La función `putchar` tiene el mismo efecto que `printf` si solo se imprime un caracter con "%c"

La sentencia

```
putchar(65);
```

Tiene el mismo efecto que la sentencia

```
printf("%c", 65);
```

# Función putchar

# Función putchar

Además de la diferente complejidad de las sentencias, la diferencia fundamental reside en el valor devuelto.

# Función putchar

Además de la diferente complejidad de las sentencias, la diferencia fundamental reside en el valor devuelto.

La función `putchar` devuelve el valor entero del carater impreso.

# Función putchar

Además de la diferente complejidad de las sentencias, la diferencia fundamental reside en el valor devuelto.

La función `putchar` devuelve el valor entero del carater impreso.

La función `printf` devuelve la cantidad de caracteres impresos.

# Función getchar

# Función getchar

Se puede ingresar cualquier caracter desde el teclado utilizando la función `getchar`

# Función getchar

Se puede ingresar cualquier caracter desde el teclado utilizando la función `getchar`

`getchar` devuelve un entero correspondiente al caracter ingresado

# Función getchar

# Función getchar

```c
#include <stdio.h>
// u3-getchar.c

int main (void)
{
  int numero;

  printf("Ingrese un caracter de la tabla ASCII: ");
  numero = getchar();

  printf("En la tabla ASCII es el %d\n", numero);

  return 0;
}
```

# Función getchar

```c
#include <stdio.h>
// u3-getchar.c

int main (void)
{
  int numero;

  printf("Ingrese un caracter de la tabla ASCII: ");
  numero = getchar();

  printf("En la tabla ASCII es el %d\n", numero);

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u3-getchar.c
$
```

# Función getchar

```c
#include <stdio.h>
// u3-getchar.c

int main (void)
{
  int numero;

  printf("Ingrese un caracter de la tabla ASCII: ");
  numero = getchar();

  printf("En la tabla ASCII es el %d\n", numero);

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u3-getchar.c
$ ./a.out
```

# Función getchar

```c
#include <stdio.h>
// u3-getchar.c

int main (void)
{
  int numero;

  printf("Ingrese un caracter de la tabla ASCII: ");
  numero = getchar();

  printf("En la tabla ASCII es el %d\n", numero);

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u3-getchar.c
$ ./a.out
Ingrese un caracter de la tabla ASCII:
```

# Función getchar

```c
#include <stdio.h>
// u3-getchar.c

int main (void)
{
  int numero;

  printf("Ingrese un caracter de la tabla ASCII: ");
  numero = getchar();

  printf("En la tabla ASCII es el %d\n", numero);

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u3-getchar.c
$ ./a.out
Ingrese un caracter de la tabla ASCII: A
```

# Función getchar

```c
#include <stdio.h>
// u3-getchar.c

int main (void)
{
  int numero;

  printf("Ingrese un caracter de la tabla ASCII: ");
  numero = getchar();

  printf("En la tabla ASCII es el %d\n", numero);

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u3-getchar.c
$ ./a.out
Ingrese un caracter de la tabla ASCII: A
En la tabla ASCII es el 65
$
```

# Operador de Conversión de tipo (cast)

# Operador de Conversión de tipo (cast)

En ocaciones se necesita obtener resultados de un tipo de datos a partir de variables de tipos diferentes.

# Operador de Conversión de tipo (cast)

En ocaciones se necesita obtener resultados de un tipo de datos a partir de variables de tipos diferentes.

Por ejemplo un promedio (punto flotante) a partir calificaciones (enteras)

# Operador de Conversión de tipo (cast)

En ocaciones se necesita obtener resultados de un tipo de datos a partir de variables de tipos diferentes.

Por ejemplo un promedio (punto flotante) a partir calificaciones (enteras)

```
promedio = suma_notas / cuantas_notas;
```

# Operador de Conversión de tipo (cast)

# Operador de Conversión de tipo (cast)

```c
#include <stdio.h>
// u3-sin-cast.c

int main (void)
{
  int suma_notas, cuantas_notas;
  float promedio;

  printf("Ingrese la suma de todas las notas: ");
  scanf("%d", &suma_notas);
  printf("Ingrese cuantas notas son: ");
  scanf("%d", &cuantas_notas);

  promedio = suma_notas / cuantas_notas;

  printf("Promedio: %.2f\n", promedio);

  return 0;
}
```

# Operador de Conversión de tipo (cast)

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-sin-cast.c
$
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-sin-cast.c
$ ./a.out
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-sin-cast.c
$ ./a.out
Ingrese la suma de todas las notas:
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-sin-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-sin-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
Ingrese cuantas notas son:
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-sin-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
Ingrese cuantas notas son: 2
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-sin-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
Ingrese cuantas notas son: 2
Promedio: 6.00
$
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-sin-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
Ingrese cuantas notas son: 2
Promedio: 6.00
$
```

El problema es que cuando la división es entre dos enteros, se realiza de la manera básica, devolviendo un entero y dejando resto...

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-sin-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
Ingrese cuantas notas son: 2
Promedio: 6.00
$
```

El problema es que cuando la división es entre dos enteros, se realiza de la manera básica, devolviendo un entero y dejando resto...

...asignando un valor entero a la variable `promedio` (aunque quede almacenado como de punto flotante)

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-sin-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
Ingrese cuantas notas son: 2
Promedio: 6.00
$
```

El problema es que cuando la división es entre dos enteros, se realiza de la manera básica, devolviendo un entero y dejando resto...

...asignando un valor entero a la variable `promedio` (aunque quede almacenado como de punto flotante)

Para esto se usa el operador de conversión de tipo

# Operador de Conversión de tipo (cast)

# Operador de Conversión de tipo (cast)

El operador de conversión de tipo o simplemente *cast* es un operador **unario** que cambia temporalmente el tipo de datos de su operando.

# Operador de Conversión de tipo (cast)

El operador de conversión de tipo o simplemente *cast* es un operador **unario** que cambia temporalmente el tipo de datos de su operando.

Consiste en colocar entre parentesis el tipo de datos al que se quiere *convertir* el operando, delante del mismo.

# Operador de Conversión de tipo (cast)

# Operador de Conversión de tipo (cast)

```c
#include <stdio.h>
// u3-con-cast.c

int main (void)
{
  int suma_notas, cuantas_notas;
  float promedio;

  printf("Ingrese la suma de todas las notas: ");
  scanf("%d", &suma_notas);
  printf("Ingrese cuantas notas son: ");
  scanf("%d", &cuantas_notas);

  promedio = (float) suma_notas / cuantas_notas;

  printf("Promedio: %.2f\n", promedio);

  return 0;
}
```

# Operador de Conversión de tipo (cast)

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-con-cast.c
$
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-con-cast.c
$ ./a.out
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-con-cast.c
$ ./a.out
Ingrese la suma de todas las notas:
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-con-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-con-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
Ingrese cuantas notas son:
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-con-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
Ingrese cuantas notas son: 2
```

# Operador de Conversión de tipo (cast)

```
$ gcc -Wall -std=c99 -pedantic-errors u3-con-cast.c
$ ./a.out
Ingrese la suma de todas las notas: 13
Ingrese cuantas notas son: 2
Promedio: 6.50
$
```

# Precedencia (actualizada)

# Precedencia (actualizada)

| Operador | Asociatividad |
|---|---|
| $()$ | Izq. a Der. |
| $+ \quad - \quad (\text{tipo})$ | Der. a Izq. |
| $* \quad / \quad \%$ | Izq. a Der. |
| $+ \quad -$ | Izq. a Der. |
| $< \quad <= \quad > \quad >=$ | Izq. a Der. |
| $== \quad !=$ | Izq. a Der. |
| $=$ | Der. a Izq. |