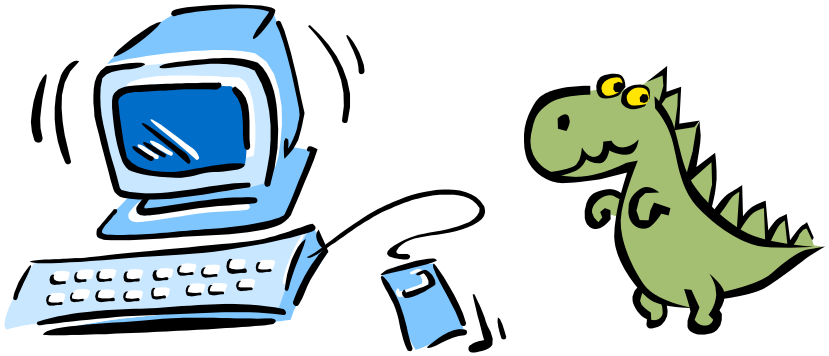
El Lenguaje C



Facultad de Cs. Empresariales
Depto. Sistemas de Información
Universidad del Bio-Bio

1.- Introducción

- Diseñado por Brian Kernighan y Dennis Ritchie a mediados de los años 70.
- Es una lenguaje de alto nivel que posee capacidades de los lenguajes de bajo nivel.
- C es un lenguaje muy poderoso cuyo desarrollo está estrechamente vinculado al sistema operativo UNIX.

2.- Variables y tipos de datos

La variables representan objetos (datos) que son manipulados por los los programas. Una variable tiene asociado un *nombre*, un *tipo* y un *valor*.

Tipos de datos en C

int : Para números enteros

char : Para caracteres

float : Para números reales de precisión simple (aprox. 7 decimales)

double : Para números reales de precisión doble (aprox. 16 decimales)

◆Para que una variable pueda ser utilizada debe ser declarada en el programa. La declaración se realiza:

<tipo> <nombre>;

Por ejemplo:

int a ; */* la variable a es de tipo entero */*

char c,d ; */* las variables c y d pueden almacenar un carácter * /*

float temperatura ; */* la variable temperatura es de tipo float */*

3.- Asignación y expresiones

- La asignación consiste en otorgar un valor "*legal*" a una variable. Las asignaciones en C son del tipo:

<Variable> = <expresión>

donde

<Variable> : *puede ser cualquier variable y*

<expresión> : *Puede ser un valor, una expresión u otra variable que asignen un valor válido para el tipo de datos de la variable.*

Asignaciones

Ejemplo:

```
int p,q ;      /* declara variables p y q de tipo entero */
char c ;       /* declara variable c como carácter   */
p = 23 ;       /* asigna el valor 23 a p     */
q = p ;        /* asigna el valor de p a la variable q */
 c = “s”       /* asigna el carácter “s” a la variable c */
```

```
float temperatura = 0,0004 ;    /* también es permitido */
```

Expresiones

Expresiones aritméticas

Se construyen utilizando variables, constantes, paréntesis y los siguientes operadores:

suma	+
resta	-
multiplicación	*
división	/
resto de división	%

Ejemplo:

i = 6+4

j = (a * (5/f))+ (a-f)

Expresiones Lógicas

Entregan un valor *Verdadero* o *Falso*. En lenguaje C una expresión entrega un valor 0 si es Falso y un valor distinto de 0 (comúnmente 1) si es Verdadero.

Las expresiones lógicas se construyen a partir de operadores de relación como:

> , < , >= , <= , = = (igual) , ! = (distinto)

Las expresiones lógicas más complejas se construyen utilizando los conectivos lógicos:

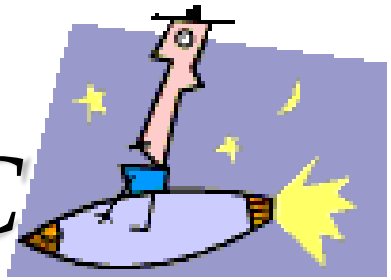
&& : AND || : OR ! : NOT

4.- Plantilla general de un programa en C

```
#include <stdio.h>        /* declaración de importaciones */
#define  b  2             /* declaración de constantes  */
int a;                    /* declaración de variables  */


main()                    /* procedimiento principal  */
{                         /* inicio de programa  */
a = b * 5;
printf("%d",a);
}                         /* fin de programa  */
```

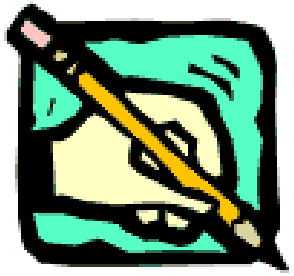
5.- Programas en C



- Un programa en C se conforma por un conjunto de funciones (procedimientos) que retornarán algún resultado.
- Por ejemplo, la función *main()* es la función principal de un programa, y es la primera "parte" del programa que se ejecuta.
- Algunas funciones pueden ser creadas por el programador, así como también pueden usarse otras disponibles.
- Pero...¿ Cómo se pueden utilizar estas funciones ya creadas ?

  … Esto se realiza por medio de las **librerías**.

6.- Librerías



- La librerías contienen funciones ya creadas que podemos utilizar (*concepto de reuso*) . Estas funciones permiten desde sumar dos números hasta imprimir un conjunto de cacateres o ingresar datos desde algún periférico.

- Cada librería tiene un conjunto definido de funciones. Por ejemplo, la librería estándar de C, "***STDIO.H***" contiene funciones que permiten imprimir valores en la pantalla, recibir datos desde el teclado o realizar operaciones trigonométricas.

- Para indicar que se va a utilizar alguna librería se debe indicar mediante la cláusula ***#INCLUDE***.

Librería Estándar <stdio.h>

- La función PRINTF permite imprimir valores en la pantalla. Esta función recibe como primer parámetro el objeto a ser impreso.

  Por ejemplo : printf(“ Esto aparecerá en pantalla”)

  entregará como resultado en la pantalla

  Esto aparecerá en pantalla

- Se pueden incorporar marcas que indicarán donde insertar valores que aparecerán en los siguientes parámetros del printf. Por ejemplo, para imprimir el valor de la variable suma :

  printf(“ La suma es : %d “,suma);

Librería Estándar <stdio.h>

- Estas “marcas” permiten imprimir valores y existen marcas para distintos tipos de datos:
  - **%d** para imprimir un entero
  - **%c** para imprimir un carácter
  - **%f** para imprmir un float, etc.

- Por ejemplo: Si las variables int i=8, y float j=5,3 se desean imprimir debería utilizarse:

```
printf(“El valor de i es %d y el valor de j es %f .”,i, j)
```

lo que entrega por pantalla

```
El valor de i es 8 y el valor de j es 5,3
```

Librería Estándar <stdio.h>

- La función **SCANF** permite leer un dato desde el teclado y asociarlo a una variable. Para esto se utilizan también marcas de acuerdo al tipo de la variable.

- Por ejemplo para recibir un valor desde el teclado para la variable entera i escribimos:

scanf(“%d”,&i)

donde : “%d” , indica el formato de entrada

&i , indica la variable a la cual se va asociar el valor ingresado.

7.- Programa ejemplo

- El siguiente programa permite que el usuario ingrese valores a las variables a y b (utilizando la función *scanf*) y luego los muestra por pantalla (utilizando la función *printf*)

Obs: Nótese que la función *scanf* muchas veces es complementada con la función *printf*, para orientar al usuario acerca del datos que se debe ingresar.

Programa ejemplo

```
#include <stdio.h>
int a,b;
main()
{
printf(“Ingrese un valor para a  \n”);
scanf(“%d”,&a);
printf(“ \n Ingrese un valor para b  \n”);
scanf(“%d”,&b);
printf(“\n El valor de a es %d y el valor de b es %d”,a,b);
}
```

8.- Estructura IF

- Permite ejecutar alguna acción dependiendo del resultado de una comparación. Si el resultado de la comparación es distinto de 0 se realiza el primer conjunto de acciones, de los contrario se ejecuta el segundo conjunto de acciones.
- La forma general de esta estructura de selección es la siguiente:

```
IF expresión        {
                    <primer grupo de acciones>
                    }
            ELSE        {
                        <segundo grupo de acciones>
                        }
```

9.-*Estructura WHILE*

- Esta estructura repite un conjunto de instrucciones *mientras* se cumpla una expresión lógica sea verdadera (o sea retorne un valor distinto de 0).
- Su formato general es: ||

```
WHILE  expresión
       {
        <conjunto de acciones>
        }
```

10.- Estructura DO…WHILE

- Esta estructura repetitiva comprueba si se cumple o no la condición al final de ejecutar el conjunto de instrucciones.
- Su funcionamiento puede entenderse como: “Realizar el conjunto de instrucciones, mientras se cumpla la condición”.
- Su formato general es el siguiente:

```
DO {
        <conjunto de acciones>
        }
WHILE expresiones
```

11.- Estructura FOR

- La estructura FOR permite iterar utilizando una variable desde una *valor inicial* hasta una *valor final* que se afecta por un *incremento* que se indica.
- Su formato general es:

```
FOR (var=valor_inicial ; condición ; incremento_de_var)
    {
     <conjunto de acciones>
     }
```

12.- Del Algoritmo al Programa

Problema: Generar los n primeros números pares.

(Utilizando estructura *"Repita ... Hasta que"* del Pseudocódigo)

Algoritmo *Números pares*

Variables i , par , n : **entero**
Inicio
leer n
i <-- 0
Repita
    i <-- i + 1
    par <-- 2 * i
    Mostrar par
Hasta que (i=n)
Fin

Del Algoritmo al Programa

Problema: Generar los n primeros números pares.

Programa en Lenguaje C

```
#include <stdio.h>
int n, i , par;
main()
{
i=0;
printf(“ Cantidad de numeros pares : \n”);
scanf(“%d”,&n);
do {
  i++; /* similar a usar i = i+1 */
  par=2*i;
  printf(“ %d”, par);
   } while ( i < n);
}
```