LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

UNIVERSIDAD NACIONAL DE TUCUMÁN
facet
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

PROGRAMACIÓN

Unidad 3: Concepto de Tipos de Datos. Tipos de Datos Simples. Constantes y variables. Entrada y Salida básicas. Estructuras de Selección. Construcciones de Selección múltiple. Sintaxis y semántica. Estructuras de Control Iterativas. Sintaxis y semántica.

Lic. Mariela A. Velázquez

Concepto de Tipos de Datos

- Los lenguajes de alto nivel permiten hacer abstracciones e ignorar los detalles de la representación interna, usando el concepto de tipo de datos. Así la información almacena en memoria no es tratado como una secuencia anónima de bits, sino como valores enteros, reales, carácter.

- Podemos clasificar los tipo de datos en dos grandes grupos: datos simples y datos compuestos

Tipos de Datos Simples

- Los especificadores de tipo de dato, determinan el tipo de dato o elemento de información que contendrá el área de memoria reservada para las variables declaradas por medio de la declaración en proceso. Las palabras clave que constituyen sus correspondientes especificadores , son los siguientes:

Tipo	Palabra reservada
Carácter	char
Entero de longitud estándar	int
Entero de longitud grande	Long int
Entero de longitud pequeña	Short int
Entero sin signo	unsigned
Punto flotante (real)	float
Punto flotante doble precisión	double

Analizamos una estructura de solución propuesta.

- El código en C encierra varios conceptos:
  - Variables.
  - Declaraciones.
  - Asignaciones.
  - Expresiones aritméticas.
  - Salida con formato.

Variables

- Una variable es un objeto cuyo valor cambia durante la ejecución del programa, que tiene un **nombre** y ocupa una cierta posición de memoria.
- Todas las variables que se usan deben ser declaradas. La idea de ello es que se le avisa al compilador de las propiedades de la variable en cuestión. Esto es de los valores que pueden asumir y de las operaciones permitidas.

Declaración de una Variable

- La declaración se hace usando una **palabra reservada** del lenguaje.
- En el ejemplo, **int** es la palabra reservada, que le avisa al compilador que ha de trabajar con enteros.
- Una palabra reservada es una palabra con significado especifico, predefinido por el lenguaje que se usará.

Constantes y tipo asociado

- A definir constantes de tipo numérico se puede indicar el calificador asociado.
- Un entero long se escribe con una l o L terminal
- Las constantes sin signo con u o U.
- También se pueden definir constantes de carácter y de cadena.

Entrada y salida básicas

- C no contiene instrucciones de entrada ni de salida.
- El lenguaje C interpreta que la entrada proviene de stdin(dispositivo estándar de entrada).
- La salida es dirigida a la stdout(dispositivo estándar de salida).
- Ambas pueden redirigirse a otros dispositivos.

Función de Salida: printf

- Permite la presentación de valores numéricos, caracteres y cadenas de texto por el archivo estándar de salida (pantalla) .El prototipo de la función printf es el siguiente:

printf(control,arg1,arg2…);

- **Control:** la cadena de control en la cual se indica la forma en que se mostrarán los argumentos posteriores (si los hubiere) .También se puede escribir una cadena de texto (sin necesidad de argumentos), o combinar ambas posibilidades, así como secuencias de escape.

Función de Salida: printf

printf(control, arg_1, arg_2...);

arg_i: argumentos cuyos valores habrán de ser mostrados en la línea de salida. Si se utilizan argumentos se debe indicar en la cadena de control tantos caracteres de conversión (o modificadores) como argumentos se van a presentar.

El formato completo de los modificadores: % [signo] [longitud] [.precisión] [l/L] conversión

Entre el % y el carácter de conversión puede haber:

- **Signo:**

  - : indica si el valor se ajustará a la izquierda.

  +: establece que el número siempre será impreso con signo.

- **longitud:** especifica un ancho mínimo de campo. El argumento será impreso en por lo menos esta longitud. Si tiene menos caracteres que el ancho del campo, será rellenado a la izquierda. El carácter de relleno normalmente es espacio.

- **precisión:** indica el número máximo de decimales que tendrá el valor.

- **l/L:** utilizamos l cuando se trata de una variable de tipo long y L cuando es de tipo double.

Listado de los modificadores o caracteres de conversión más utilizados

- %c Un único carácter
- %d Un entero con signo, en base decimal
- %u Un entero sin signo, en base decimal
- %o Un entero en base octal
- %x Un entero en base hexadecimal
- %e Un número real en coma flotante, con exponente
- %f Un número real en coma flotante, sin exponente
- %s Una cadena de caracteres
- %p Un puntero o dirección de memoria

Secuencias de Escape

Nombre	Secuencia de Escape
Nulo	\o
Retroceso	\b
Tabulador	\t
Salto de línea	\n
Salto de página	\f
Retorno de carro	\r
Comillas	\"
Signo de interrogación	\?
Barra invertida	\\

Las secuencias de escape permiten expresar ciertos caracteres no imprimibles, así como la barra inclinada hacia atrás (\) y la comilla simple (´) a través de la combinación adecuada de algunos caracteres alfabéticos. Una secuencia de escape siempre comienza con una barra inclinada hacia atrás (barra invertida) y es seguida por uno o más caracteres especiales. Por ejemplo, un salto de línea (LF), que representa el carácter de nueva línea, se representa con un \n. Las secuencias de escape son interpretadas por el compilador como un solo carácter.

Función de Entrada: scanf

Función scanf(): Permite ingresar la información o datos en posiciones de memoria de la computadora a través del archivo estándar de entrada (teclado).El prototipo de la función scanf es el siguiente:

scanf(control,arg1, arg2…);

Control:cadenadecontrolqueindica,porreglageneral,losmodificadoresqueharánreferenciaaltipodedatodelosargumentos.

Función de Entrada: scanf

scanf(control,arg_1, arg_2...);

arg_i: son los nombres o identificador es de los argumentos de entrada cuyos valores se incorporarán por teclado.

Scanf "necesita conocer" la posición de la memoria en que se encuentra la variable para poder almacenar la información ingresada. Por eso se utiliza un operador unario de dirección: **&(ampersand)**, que se coloca delante del nombre de cada variable.

scanf(“%d %d %d”, &var1, &var2, &var3);

Si Ud. no indica a **scanf()** cual es la dirección interna (la posición de memoria) de estas variables, el programa en cuestión no podría acceder luego a los valores tipiados desde el teclado.

Leer y escribir un carácter

Con formato %c, en una invocación a la función scanf.

```
char letra;
scanf("%c",&letra);
```

```
Letra=getchar();
putchar(letra);
```

Con formato %c, en una invocación a la función printf.

```
char letra;
printf("%c",letra);
```

Estructura de selección

Hasta ahora los programas se ejecutaban por un secuencia ordenada de proposiciones, cada una de ellas se ejecuta una vez o un número fijo de veces.

Cuando estudiaron diseño vieron la estructura Si-sino:

SI condición ENTONCES

Acción 1;

SINO

Acción 2;

Estructura de selección

En el lenguaje C se dispone de sentencias condicionales: if-else, switch que permite seleccionar una única alternativa entre varias para ser ejecutada.

Sentencia if-else

Esta sentencia permite seleccionar el grupo de sentencias o bloques que serán ejecutadas, de acuerdo al valor de una condición.

Formalmente, la expresión:

```
if(expresión)
{
   proposición 1;
}
else{
        proposición 2;
    }
```

Sentencia if-else

Expresión: es una expresión que tiene que ir entre ( ) y tiene que ser evaluada como VERDADERA (distinto de cero) o FALSO (CERO).

Proposición: puede ser simple o compuesta { }.

La semántica de una sentencia **if** es:

- **Con else**: se evalúa la expresión, si es verdadera, se lleva a cabo la proposición 1, sino si es falsa, se ejecuta la proposición 2.
- **Sin else:** se evalúa la expresión, si es verdadera se ejecuta la proposición 1

if anidados

```
if (expresión 1)
      proposición 1;
else
          if (expresión 2)
              proposición 2;
          else
               if (expresión 3)
                       proposición 3;
               else
```

Es la forma general de escribir una decisión múltiple.

Las expresiones se evalúan en orden, si cualquier expresión es verdadera, se ejecuta la proposición asociada a ella.

Sentencia Switch

```
SEGÚN variable
        const 1: A1;
        const 2: A2;
                ...
SINO
        An
```

```
Switch(expresión)
{
        case const1: sent1;
                        break;
        case const2: sent2;
                        break;
                ...
        default: sentencias;
}
```

Sentencia Switch

Expresión: la expresión puede ser una expresión entera, un carácter alfanumérico, una constante o una variable.

Const.: puede ser una expresión entera, un carácter alfanumérico, una constante o una variable.

Sent.: puede ser cualquier bloque de sentencias de C. Si el bloque consta de una sola sentencia no es necesario incluir las llaves de apertura y cierre. De igual manera se recomienda incluirlas.

default (case predeterminado): es opcional. En general se inserta la línea default al final del cuerpo de switch cuando se estima que el case predeterminado va a ocurrir con mayor frecuencia.

Break: esta sentencia provoca una salida inmediata del switch. Puesto que los case sirven sólo como etiquetas, después de que se ejecutan el código para uno el código pasa al siguiente. Las formas más comunes de dejar un switch son **break** y **return**

Sentencia While

La sentencia while ejecuta una sentencia, simple o compuesta, cero o más veces, dependiendo de una condición.

Su sintaxis es:

while (condición) sentencia ;

Donde:

condición es cualquier expresión numérica, relacional o lógica.

sentencia es una sentencia simple o compuesta.

Sentencia While

while (condición) sentencia ;

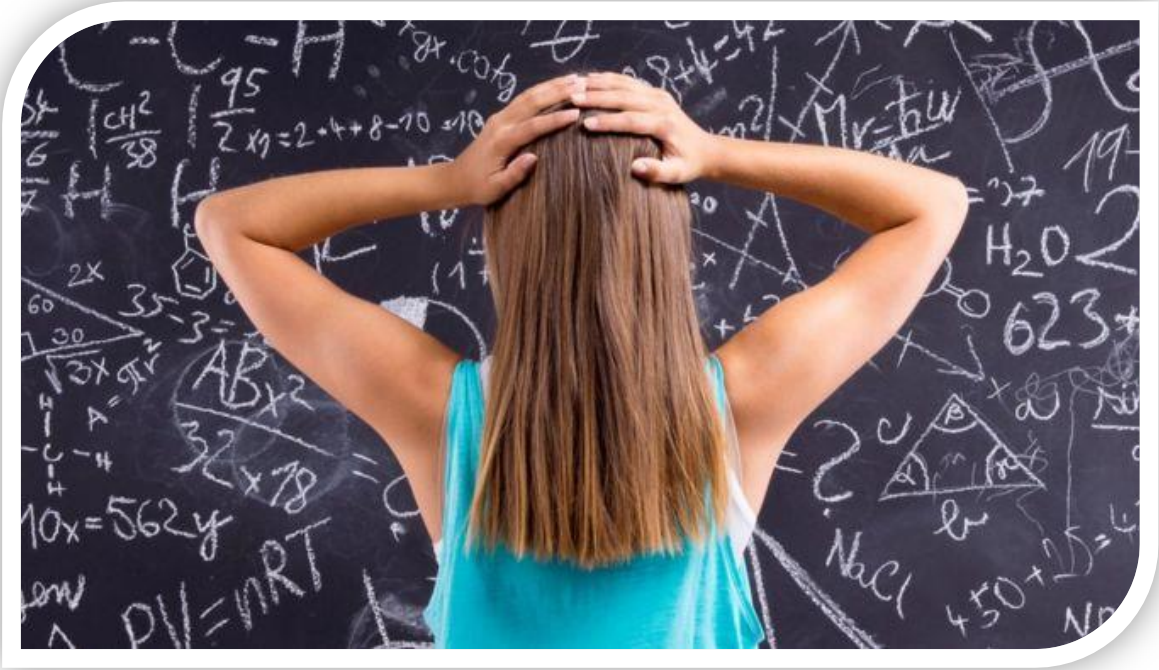
La ejecución de la sentencia while sucede así:

Se evalúa la condición . Si el resultado de la evaluación es 0 (falso), la sentencia no se ejecuta y se pasa el control a la siguiente sentencia en el programa. Si el resultado de la evaluación es distinto de 0 (verdadero), se ejecuta la sentencia y el proceso descrito se repite desde el Inicio.

Ejercicio

Diseñar un algoritmo que muestre la tabla de multiplicar de un número determinado.

Además se solicita que muestre por pantalla hasta que número se desea ver la tabla. Implemente en C.



Sentencia do-while

En el lazo do-while tiene la comprobación relacional al final, en vez de tenerla al inicio como es en la estructura while. La sintaxis es:

```
do {
        sentencia;
}while ( condición );
```

La sentencia se ejecuta y después se evalúa la condición. Si es verdadera la proposición se evalúa de nuevo. Cuando la expresión se hace falsa el ciclo termina.

Sentencia do-while

- Retomar el ejemplo anteriormente realizado usando la estructura do-while.

- ¿Qué diferencias nota entre while y do-while?.

Sentencia For

A ver Marta, pase y escriba 500 veces no debo tirar papeles en clases...

```
#include <stdio.h>

int main()
{
    int cont;

     for(cont=1;  cont<=500;  cont++)
    {
        printf("no debo tirar papeles en clases");
     }
     return 0;
}
```

desmotivaciones.es

Cosas de programadores.

Sentencia for

La sentencia for permite ejecutar una sentencia simple o compuesta, repetidamente un número de veces conocido.

Su sintaxis es la siguiente:

for (expresión-comienzo; condición; progresión-condición)

Sentencia;

Expresión-comienzo: representan variables de control que serán iniciadas con los valores de las expresiones.

Condición: es una expresión booleana que si se omite, se supone verdadera.

Progresión-condición: es una o más expresiones separadas por comas cuyos valores evolucionan en el sentido de que se cumpla la condición para finalizar la ejecución de la sentencia for.

Sentencia: es una sentencia simple o compuesta.

```c
#include <stdio.h>

int main()
{
    printf("Hasta la próxima clase!!\n");

    return 0;
}
```