Laboratorio I

Tipos de Datos – Expresiones
Variables - Constantes

FECyT
Facultad de Ciencias
Exactas y Tecnológicas

UNSE
Universidad Nacional
de Santiago del Estero

Tipos de Datos

- Un **tipo de dato** especifica el **conjunto de valores** o dominio de valores que puede tomar una variable; las **operaciones básicas permitidas**, estableciendo además la **cantidad de memoria** necesaria para almacenar una variable de ese tipo.

- El lenguaje C posee el siguiente **conjunto de tipos de datos**:
  - **int**: permite almacenar un numero entero.
  - **char**: puede contener un carácter del conjunto de caracteres
  - **float**: permite almacenar un numero en coma flotante
  - **double**: puede contener número en coma flotante de precisión doble

  - Se dice que C es un lenguaje **fuertemente tipado.**

Laboratorio I – Versión Preliminar - Aldo Roldán

Variables

- Una **variable** es una porción de memoria la cual consta de uno o más bytes contiguos.
- Toda variable tiene un **nombre**, y se debe utilizar ese nombre para referirse a ese lugar de la memoria.
- Además, como su nombre lo indica**, el valor de una variable** no es fijo, y **puede cambiar** a lo largo del programa tantas veces como resulte necesario.
- **Existen diferentes tipos de variables**, y cada tipo de variable es utilizado para almacenar una clase particular de datos. Así existen variables enteras, reales, etc. las cuales analizaremos posteriormente.

Laboratorio I – Versión Preliminar - Aldo Roldán

Variables

- De manera general la **declaración de variables** tendrá el siguiente formato:

```
tipo_de_dato  nombre_var_1[ , nombre_var_2[ , .....
    nombre_var_n];
```

- Esta declaración a su vez es una **definición de una variable** debido a que la misma causa además que se asigne la cantidad de memoria necesaria para almacenar un valor.
- **Ejemplos**:

```
int numero;
char letra = 'a';
int contA, contB, contC;
```

Laboratorio I – Versión Preliminar - Aldo Roldán

Nombres de Variables

- El **nombre de la variable**, será una secuencia de uno o mas letras minúsculas o mayúsculas, dígitos y el carácter subrayado (_) que comienza con una letra (incidentalmente, el carácter subrayado cuenta como una letra).
- Un punto importante a recordar es que los nombres de las variables son **sensibles al contexto**, es decir existe diferencia entre mayúsculas y minúsculas; siendo la variable Saldo distinta de saldo.
- Otro aspecto a recordar, es que **siempre se deben declarar las variables antes de su uso**, de lo contrario el compilador no la reconocerá y dará error.
- Por último, **las palabras reservadas del lenguaje no se pueden usar** como nombres de variable

Laboratorio I – Versión Preliminar - Aldo Roldán

Operador de Asignación

- El **símbolo =** se denomina **operador de asignación** debido a que asigna el valor de la derecha a la variable de la izquierda.
- Por lo general, es mucho mejor inicializar la variable cuando se la declara, de manera que se elimina la duda acerca de si fue inicializada o no una variable.
- **Ejemplos:**

```
int numero = 10;
char letra = 'a';
int contA = 0, contB = 0, contC = 0;
```

Laboratorio I – Versión Preliminar - Aldo Roldán

Enteros

- Se tienen **cinco tipos** de variables que se pueden declarar para almacenar **valores enteros con signo**
-

Nombre del Tipo	Numero de Bytes	Rango de Valores
short int	2	-32768 a +32767
int	4	-2147483648 a +2147483647
long int	4	-2147483648 a +2147483647

- Los nombres de tipo **short** y **long** pueden ser utilizados como abreviaturas para los nombres de tipo **short int** y **long int**
- La tabla de arriba refleja el tamaño típico de cada tipo de dato entero, aunque dependen del compilador utilizado.

Laboratorio I – Versión Preliminar - Aldo Roldán

Enteros

- Para cada uno de los tipos que almacenan enteros con signo, existe un tipo correspondiente que almacena **enteros sin signo**, a los que se le antepone la palabra clave **unsigned**.

Nombre del Tipo	Numero de Bytes	Rango de Valores
unsigned short int	1	0 a 65535
unsigned int	2	0 a 4294967295
unsigned long int	4	0 a 4294967295

Laboratorio I – Versión Preliminar - Aldo Roldán

Constantes

- Debido a que se tienen diferentes clases de variables enteras, puede esperarse tener distintas clases de **constantes enteras**.
- Si desea asegurarse que es de tipo long, debe agregarle una **letra L en mayúscula o en minúscula** al final del valor numérico.
- Una constante entera también será de tipo long por defecto si se encuentra fuera del rango del tipo int.
- Para especificar que una constante es de un tipo sin signo, debe agregársele una U.
- **Ejemplos:**

```
long numeroGrande = 125457963L;
unsigned int cantidad = 100U;
unsigned long valor = 9999999999UL;
```

Laboratorio I – Versión Preliminar - Aldo Roldán

Expresiones Aritmetica

- Una **expresión aritmética** especifica un calculo usando valores almacenados en variables y/o constantes los cuales son combinados utilizando operadores aritméticos .
- Una expresión que resulta en un valor numérico se dice que es una **expresión aritmética.**
- **Ejemplos:**

```
3 + 5 * 3
2019 - anioNacimiento
```

- Evaluar cualquier expresión, produce un solo valor numérico

Laboratorio I – Versión Preliminar - Aldo Roldán

Operadores

- Un **operador** es un símbolo (+, -, *, /, etc) que tiene una función predefinida (suma, resta, multiplicación, etc) y que recibe sus argumentos de manera infija, en el caso de tener 2 argumentos (**a operador b**), o de manera prefija o postfija, en el caso de tener uno solo (**operador a** , o bien, **a operador**).
- **En C existen una gran variedad de operadores**, que se pueden agrupar de la siguiente manera:
  - Operadores aritméticos
  - Operadores relacionales
  - Operadores lógicos
  - Operadores a nivel de bit (bitwise operators)
  - Operadores especiales

Operadores Aritméticos

Operador	Acción	Ejemplo
+	Suma	n = 2 + 3; /* n toma el valor 5*/
-	Resta	n = 7 - 3; /* n toma el valor 4*/
*	producto	n = 2 * 3; /* n toma el valor 6*/
/	División	n = 7 / 3; /* n toma el valor 2*/
%	Modulo o resto	n = 7 % 3; /* n toma el valor 1*/
++	Incremento	n = 7; n++ /* n toma el valor 8*/
--	Decremento	n = 7; n-- /* n toma el valor 7*/

Precedencia de Operadores

La siguiente tabla lista todos los operadores desde la precedencia más alta a la más baja.

Operador	Descripción
++ --	Incremento, decremento
* / %	Multiplicación, división y modulo
+ -	Suma y resta
>> <<	Desplazamiento a nivel de bits a derecha e izquierda
<= < > >=	Operadores de comparación
== !=	Operadores de igualdad
&	Y a nivel de bits
^ |	O exclusivo y O regular a nivel de bits

Laboratorio I – Versión Preliminar - Aldo Roldán

Precedencia de Operadores

Operador	Descripción
&&	Y lógico
||	O lógico
= -= += *= /= %=	Operadores de asignación

- La **asociatividad** ayuda a determinar el orden de las operaciones cuando nos encontramos con dos o mas operadores de idéntica precedencia.
- Casi la mayoría de los operadores tienen asociatividad de **izquierda a derecha**.

Laboratorio I – Versión Preliminar - Aldo Roldán

Mostrando valores enteros

- La forma más sencilla de mostrar valores es utilizar la funcion **printf**.
- **Ejemplo**:

```
edadActual = 18;
aniosParaJubilar = 65 – 18;
printf("La edad actual es %d y le faltan %d años
parajubilarse\n", edadActual, aniosParaJubilar");
```

- La funcion utiliza tres **argumentos** o datos para llevar a cabo su tarea. El primero es un **string de control**,el cual especifica el formato en que serán mostrados los datos; y los dos ultimos son los valores que seran serán mostrados, que en este caso se obtienen a partir de las variables.

Mostrando valores enteros

- Si observa claramente, dentro del string de control vera un **%d** dentro del mismo. Este se denomina **especificador de conversión** para la variable.
- Los especificadores de conversión determinan como se muestran los valores de las variables en la pantalla. En este caso, se ha utilizado una **d** la cual es un especificador decimal que se aplica a los valores enteros.
- De forma general, los especificadores de conversión siempre comienzan con un **carácter %** de tal manera que la función printf( ) pueda reconocerlo.
- Si existieran varios especificadores de conversión, los mismos son reemplazados por los valores de las variables que aparecen como el segundo y los subsiguientes argumentos de la funcion printf( );

Mostrando valores enteros

- En el archive cabecera **limits.h** se definen los valores máximos y mínimos de los distintos tipos de datos.

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

int main(void) {
    printf("Valores Maximos/Minimos definidos en limits.h\n");
    printf("Enteros Cortos Con Signo %d %d\n", SHRT_MAX, SHRT_MIN);
    printf("Enteros Con Signo %d %d\n", INT_MAX, INT_MIN);
    printf("Enteros Largos Con Signo %ld %ld\n", LONG_MAX, LONG_MIN);
    printf("Enteros Cortos Sin Signo %u\n", USHRT_MAX);
    printf("Enteros Sin Signo %u\n", UINT_MAX);
    printf("Enteros Largos Sin Signo %u\n", ULONG_MAX);
    return (EXIT_SUCCESS);
}
```


Ingreso de información

- La función **scanf( )** es otra función que requiere la inclusión del encabezado <stdio.h>, y su proposito es el de manejar la entrada de información desde el teclado.
- **Ejemplo:**

```
printf("Ingrese su año de nacimiento:");
scanf("%d", &anioNacimiento);
printf("Ud. Tiene %d años\n", 2019-anioNacimiento);
```

- La misma posee dos argumentos:
  - el primero es **un string conteniendo un carácter de control** que indica que tipo de dato se debe leer del teclado. En el ejemplo seria **"%d"** indicando que se debe leer un valor entero; y el
  - segundo argumento es la **variable en la cual se almacenará el valor leido**, que en este caso es la variable anioNacimiento.

Ingreso de información

- Tenga en cuenta que el primer argumento es un string de control similar al utilizado en la funcion printf( ), excepto de que controla el ingreso en vez de la salida.
- Sin duda habrá notado algo nuevo: el **&** (ampersand) precediendo al nombre de la variable. Este recibe el nombre de **operador de dirección**, y es necesario para que la función **scanf ( )** pueda almacenar el valor que lee en la variable **anioNacimiento**.
- Es importante que recuerde que debe usar el **operador de dirección** (el &) antes del nombre de la variable cuando utilice la función **scanf( )**, y que no debe utilizarlo cuando usa la función **printf ( )**

Laboratorio I – Versión Preliminar - Aldo Roldán

Tipo de dato flotante

- Las variables de **punto o coma flotante** son utilizadas para almacenar números en punto flotante, los cuales contienen valores que están escritos con punto decimal, por lo cual con ellos puede representar números fraccionarios así también como números enteros.
- A menudo son expresados como un valor decimal multiplicado por alguna potencia de 10. La **E** en cada numero es el exponente y puede ponerlo si lo desea con **e** minúscula.

Valor	Con un exponente	Puede ser escrito como
1.6	0.16x10 1	0.16E1
0.00008	0.8x10-4	0.8E-4
7655.899	0.7655899x10 4	0.7655899E4

Laboratorio I – Versión Preliminar - Aldo Roldán

Tipo de dato flotante

- Existen diferentes tipos de variables de punto flotante, los cuales se muestran a continuación

Nombre del Tipo	Numero de Bytes	Rango de Valores
float	4	+-3.4E38 (6 dígitos decimales de precisión)
double	8	+-1.7E308 (15 dígitos decimales de precisión)
long double	12	+-1.19E4932 (18 dígitos decimales de precisión)

- Las variables de tipo **float** son conocidas como números de punto flotante de **simple precisión**. Las de tipo **double** son suficientes para la mayoría de los requerimientos, y le permite almacenar valores de números de punto flotante de **doble precisión, mientras que las** de tipo **long double** proporciona un rango excepcional y de precisión.

Laboratorio I – Versión Preliminar - Aldo Roldán

Declaración de variables flotantes

- Una variable de tipo punto flotante se la declara de manera similar que una variable entera, debiendo anteponerle la palabra clave del tipo que desea utilizar.
- **Ejemplos:**

```
float promedio = 3.5f;
double masGrande = 123E30;
```

- Como observara, para escribir una **constante de tipo float** deberá agregarle una **f** al numero para distinguirlo del tipo double. Esto es así, ya que se considera que un numero con punto decimal es de tipo double por defecto.

Laboratorio I – Versión Preliminar - Aldo Roldán

Impresión de variables flotantes

- Puede utilizar el especificador de formato **%f** para mostrar valores de punto flotante.
- **Ejemplos:**

```
printf("El promedio de edad es: %f\n:",promedio);
printf(El promedio de edad es: %.2f\n:",promedio);
```

- Se puede especificar la cantidad de lugares que puede ver después del punto decimal dentro del especificador de formato. Así, por ejemplo para obtener la salida de dos dígitos decimales, deberá escribir el especificador **%.2f**.
- De manera general se puede especificar el formato de un numero como: **%ancho . precisión** *modificador.*

Ingreso de variables flotantes

- Para ingresar utilizaremos la funcion scanf, solo que en este caso se utilizara el especificador de formato **%f** permitir valores de punto flotante.
- **Ejemplo:**

```
float altura;
printf("Ingrese su altura);
scanf("%f",altura);
```