LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

INGENIERÍA EN INFORMÁTICA

PROGRAMACIÓN

Unidad N° 3 – Parte 2: Tipos de datos. Operadores. Expresiones. Calificadores. Conversiones de tipos y otros operadores. Código ASCII. Estructuras de Selección : if-else, switch

2023

Lic. Mariela A. Velázquez

Repasemos lo visto

Una variable es un objeto cuyo valor cambia durante la ejecución del programa, que tiene un nombre y ocupa una cierta posición de memoria

A diferencia de las variables, las constantes mantienen su valor durante todo el programa. #define define una constante simbólica

Los lenguajes de alto nivel permiten hacer abstracciones e ignorar los detalles de la representación interna

Variables y Constantes

Conceptos de tipos de datos

C no contiene instrucciones de entrada ni de salida.

printf(control, arg1, arg2, …);

scanf(control, arg1, arg2,…);

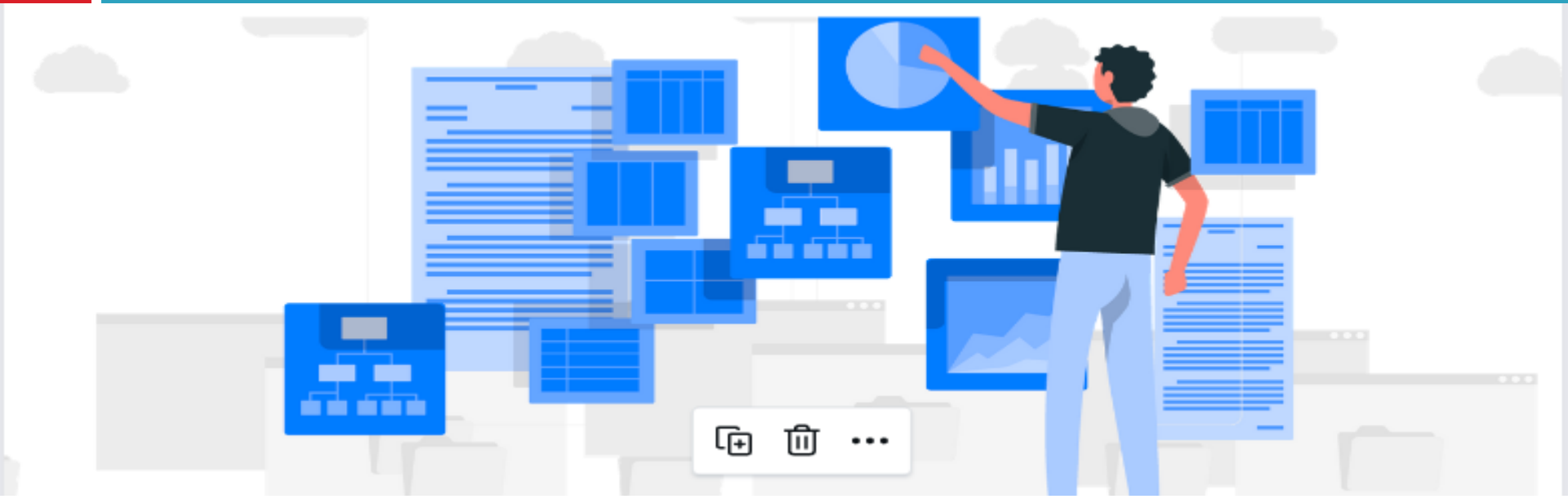
Los especificadores, determinan el elemento de información que contendrá el área de memoria reservada para las variables: int, float, char, double

Funciones de Entrada y Salida

Modificadores: %c, %d, %f, %s.

Secuencias de escape: \0, \n, \t

Tipos de Datos



Podemos definir **Dato como** una expresión general que describe los objetos con los cuales opera una computadora. A nivel de hardware, están representados por "tiras de bits" y son manejados por instrucciones propias del procesador.

El uso de **tipo de datos** es muy importante en ciencias de la computación. Un tipo identifica aquellas propiedades comunes a un grupo de objetos que lo distinguen como una clase identificable.

Tipos de datos simples

Los tipo de datos simples se caracterizan por que sus componentes no se pueden descomponer en tipos más simples. En otras palabras, representan valores atómicos. Por ejemplo: entero, carácter.

```c
#include <stdio.h>

int main() {
    printf("El tamanio de int es: %zu bytes\n", sizeof(int));
    printf("El tamanio de float es: %zu bytes\n", sizeof(float));
    printf("El tamanio de double es: %zu bytes\n",
sizeof(double));
    printf("El tamanio de char es: %zu bytes\n", sizeof(char));

    return 0;
}
```

El %zu es un especificador de para imprimir valores de tipo size_t. size_t es un tipo de datos sin signo que se utiliza comúnmente en C para representar tamaños de objetos o índices de arreglos

Tipos de dato entero: tamaño, calificadores

¿Cómo saber los valores que puede asumir una variable int?

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

int main()
{
    printf("Rango de representacion de int: %d a %d\n", INT_MIN,
INT_MAX);

    return 0;
}
```

Tipos de dato entero: tamaño, calificadores

Y surge la pregunta ¿Por qué un entero tiene limites de rango tan extraños?.

Esto tiene que ver con la forma en que la computadora almacena los números. Sabemos que los almacena en forma binaria en vez de nuestro familiar sistema decimal. El rango se calcula como $[-2^{(n-1)}$ a $2^{(n-1)} - 1$. Por ejemplo, un entero tiene 32 dígitos binarios o bits (-2^{31} a $2^{31} - 1$) de los cuales uno de ellos se usa para el signo.

Tipos de dato entero: tamaño, calificadores

Calificadores

Existen algunos calificadores que se aplican a los tipos básicos. El uso de calificadores sirve para proporcionar diferentes longitudes de enteros donde sea práctico.

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

int main()
{
    printf("Rango de representacion de int: %d a %d\n",
INT_MIN, INT_MAX);
     printf("Rango de representacion de short int: %d a %d\n",
SHRT_MIN, SHRT_MAX);
    printf("Rango de representacion de long int: %ld a %ld\n",
LONG_MIN, LONG_MAX);
    return 0;
}
```

Otros calificadores

Los números **unsigned** son siempre positivos o cero, obedecen a la aritmética módulo 2^n , con n número de bits. Los números **signed** pueden asumir valores menores a cero.

Tipo de dato float y double: Tamaño, calificadores

El tipo float es punto flotante de precisión normal .

El tipo double es punto flotante de precisión extendida.

Calificador para los tipos de datos de punto flotante

Tres tipos de punto flotante, este calificador permite trabajar con precisión extendida.

float < double < long double

Tipo de dato float y double: Tamaño, calificadores

¿Cómo saber los valores que puede asumir las variables de tipo Float?

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

int main() {

    printf("Valor minimo de float: %e\n", FLT_MIN);
    printf("Valor maximo de float: %e\n", FLT_MAX);

    return 0;
}
```

Biblioteca

Operadores

El lenguaje C es rico en operadores incorporados. Un operador es un símbolo que le indica al compilador que realice manipulaciones matemáticas o lógicas específicas. Existen cuatro clases de operadores:

Relacionales

Algoritmo	C	Descripción
>	>	Mayor
≥	>=	Mayor o igual
<	<	Menor
≤	<=	Menor o igual
=	==	Igual
≠	!=	Diferente

Lógicos

Algoritmo	C	Descripción
^	&&	And, y, conjunción
V	||	Or, o, disyunción
¬	!	Not, no, negación

Operadores

Asignación

Algoritmo	C	Descripción
←	=	Asignación

Aritméticos

Algoritmo	C	Descripción
+	+	Suma
-	-	Resta
.	*	Producto
Div - /	/	Cociente
Mod	%	Resto división entera

Prioridades de Operadores

Prioridad	Operadores
Más Alta	()
	!
	* / %
	+ -
	< <= > >=
	== !=
	&&
	||
Más Baja	= += -= *= /=

Conversiones y asignación para los tipos numéricos

En una expresión, cuando un operador tiene operandos de distintos tipos, ambos se convierten a un tipo común. En general las conversiones son de un tipo angosto a un tipo ancho sin pérdida de información.

int<float< double< long double

operando1 = operando2;

Si son de distinto tipo, el operando2 se convierte al tipo del operando 1.

i=34.567; → i=34;

```c
#include <stdio.h>
int main()
{   int entero;
    entero = 34.567;

    printf("El numero es: %d", entero);
    return 0;
}
```

El valor del punto flotante se trunca si se asigna a una variable entera.

Conversiones y asignación para los tipos numéricos

Operando1 = operando2;

Si son de distinto tipo, el operando2 se convierte al tipo del operando1.

Si j es de tipo float:

j = i + 100;

j=34 + 100; → i=134.000000;

Los enteros se convierten a flotante.

Conversiones y asignación para los tipos numéricos

Operadores de incremento y decremento

Pueden ser usados como prefijos y postfijos, incrementando antes o después de la utilización del valor. El incremento (++) o decremento (--)

Operadores de asignación y expresiones

i = i+2 es equivalente a **i += 2**

El operador de asignación es: operador =

Los operadores aritméticos binarios se asocian con un operador de asignación:

+= -= *= /= %=

```
x *= y+1;    → x = x * (y+1);
i -= g;      → i=i –g;
j %= (j –2)  → j=j % (j-2)
```

El operador ternario: ?

```c
#include <stdio.h>

int main() {

    int num1=145, num2=56, resultado;

    resultado = (num1 > num2) ? num1 : num2;

    printf("Resultado = %d", resultado);

    return 0;
}
```

Código ASCII: código estándar para el intercambio de Información

ASCII, Creado en 1963 por el Comité Estadounidense de Estándares este organismo cambio su nombre en 1969 por "Instituto Estadounidense de Estándares Nacionales" o "ANSI" como se lo conoce desde entonces.

¿Por qué hace falta la conversión?

Cada procesador de palabras codifica textos, números y los datos necesarios para el formato de márgenes, paginación, etc. En cambio, un texto en ASCII es simplemente un texto sin atributos de ninguna especie y se rige por una tabla universal.

La opción “Sólo texto” de los procesadores de palabras mas populares transforma el archivo en un texto ASCII.

Código ASCII: código estándar para el intercambio de Información

El código ASCII establece un criterio de orden:

- **Cada signo**
- **Cada número**
- **Cada letra**
- **Cada carácter de control**

} Tiene un código numérico.

Código ASCII: código estándar para el intercambio de Información

Tabla ASCII

Caracteres ASCII de control

\	barra invertida (alt + 92)
@	arroba (alt + 64)
ñ	eñe minuscula (alt + 164)
'	comilla simple, apóstrofe (alt + 39)
#	signo numeral (alt + 35)
!	signo de admiración (alt + 33)
_	guión bajo, subrayado (alt + 95)
*	asterisco (alt + 42)
~	equivalencia, tilde (alt + 126)
-	guión medio (alt + 45)

32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Caracteres ASCII imprimibles

Código ASCII: código estándar para el intercambio de Información

¿Cuál es el objetivo del uso de ASCII?

En concreto, el ASCII establece una correspondencia entre estas cadenas y los símbolos que representa (como letras y números). Esto hace que se facilite el almacenamiento y el procesamiento de información en una computadora y la comunicación entre diferentes aparatos digitales.

El uso de un código de caracteres en particular resulta extremadamente útil para evitar la pérdida de información como consecuencia de la incompatibilidad entre diferentes programas.

Tipo de dato carácter: tamaño, calificadores

Las variables de tipo char almacenan 1 solo carácter.

Declaración de una variable de tipo carácter: **char letra;**

Declaración e inicialización: **char letra ='a';**

Tamaño del tipo de dato carácter

2^n, n : número de bits.

8 bits = 1 byte

n = 1 byte

2^8=256

La tabla de caracteres ASCII básica tiene 256 caracteres.

ASCII, caracteres, valores enteros

Una constante de caracteres un entero, escrita entre ‘ ’ (comillas simples).

char letra = ‘a’; // 97 en ASCII

char digito = ‘0’; // 48 en ASCII

¿Operaciones permitidas ?

Las constantes de carácter pueden participar de operaciones aritméticas (por su “respaldo” ASCII), aunque se utilizan mas en comparaciones con otros caracteres.

Algunos caracteres se representan como constantes de cadena: \n

Son 2 caracteres que representan uno solo (Secuencias de escape).

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.

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;

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.

Ejercicio 1:

Escribir un algoritmo que lea un carácter y determine si es consonantes minúsculas o no. Usar código ASCII

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.

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

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.

Sentencia Switch

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.

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

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**

```c
#include <stdio.h>

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

    return 0;
}
```