Lógica de programación I

Juan Pablo Restrepo Uribe

Ing. Biomedico

MSc. Automatización y Control Industrial

jprestrepo@correo.iue.edu.co

2023-2

Institución Universitaria de Envigado

Evaluación

Trabajo con acompañamiento directo del docente	Trabajo independiente
• Clase magistral • Socialización por parte de los estudiantes de consultas, proyectos y trabajos realizados. • Desarrollo de ejercicios prácticos. • Trabajo guiado utilizando diferentes herramientas • Asesoría, corrección y retroalimentación del trabajo independiente	• Indagación de los recursos bibliográficos físicas y electrónicos • talleres prácticos individuales o grupales. • Prácticas de laboratorio para el desarrollo de aplicaciones • Desarrollo de actividades en las diferentes herramientas

Evaluación

Evento evaluativo	Porcentaje	Fecha
Parcial I	20 %	11 de septiembre
Parcial II	20 %	13 de noviembre
Proyecto de aula 1	15 %	05 de agosto
Proyecto de aula 2	15 %	18 de septiembre
Proyecto de aula 3	15 %	09 de octubre
Prueba de conocimientos	15 %	30 de octubre

Temáticas

Unidad	Tema	Semana
Unidad 1: Listas	Definición de nodo y lista simple Implementación de métodos insertar	1
	Implementación métodos de recorrido y de búsqueda.	2
	Implementación métodos eliminar Ejercicio de aplicación	3
	Definición lista doble e implementación de métodos de inserción	4
	Implementación de métodos de recorrido, búsqueda y eliminación	5
Unidad 2: Pilas y colas	Definición de pila e implementación de métodos (inserción, búsqueda, recorrido, eliminación)	6
	Definición de cola e implementación de métodos (inserción, búsqueda, recorrido, eliminación)	7
	Ejercicio de pilas y colas	8
Parcial	Evaluación Parcial	9
Unidad 3: Arboles y grafos	Definición de nodo binario y árbol binario	10
	Implementación métodos de insertar y recorrido	11
	Implementación métodos de borrado y ejercicios de aplicación	12
	Definición de grafo e implementación de métodos insertar y recorrido.	13
Unidad 4… Archivos	Definición de archivo: lectura y escritura desde lista	14
	Métodos de lectura y escritura desde pila, cola y árbol	15
	Ejercicios de aplicación	16
Final	Evaluación final	17

INSTITUCIÓN UNIVERSITARIA DE ENVIGADO
Ciencia , educación y desarrollo
Vigilada Mineducación

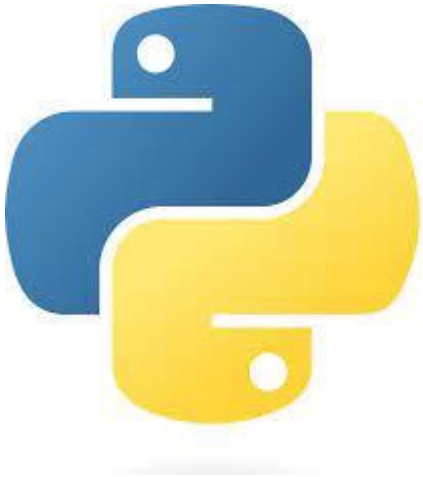
Fechas importantes

- Iniciación de Periodo Académico: El 22 de Julio de 2024
- Terminación de Periodo Académico: El 17 de Noviembre de 2024
- Iniciación de Clases: El 22 de Julio de 2024
- Terminación de Clases: El 9 de Noviembre de 2024
- Actividad Evaluativa Parcial 20%: Desde el 9 hasta el 14 de Septiembre de 2024
- Actividad de Evaluación Final 20%: Desde el 10 hasta el 17 de Noviembre de 2024
- Registro en el sistema del 60%: Hasta el 9 de Noviembre de 2024

Cursos de programación

- https://www.youtube.com/watch?v=G2FCfQj-9ig&list=PLU8oAlHdN5BlvPxziopYZRd55pdqFwkeS
- https://www.udemy.com/course/curso-python/?start=0
- https://www.youtube.com/watch?v=4PZmLUh2Z-c&list=PLoGFizEtm_6jCjWqRU8A-dQYQuo5q5KNc
- https://www.youtube.com/watch?v=5m4WORAIFr4&list=PLgHCrivozIb0TRHpUfuAUZ-VKCk8KYV_2
- https://www.youtube.com/watch?v=JJ7BMoQotEY&list=PLgHCrivozIb0ULMKfJVV-rFdRG2OeEgfq
- https://www.youtube.com/watch?v=LpIofeTqgpc&list=PLgHCrivozIb3GgqeEkDEA3j0dLGI6T6vm

Herramientas del curso

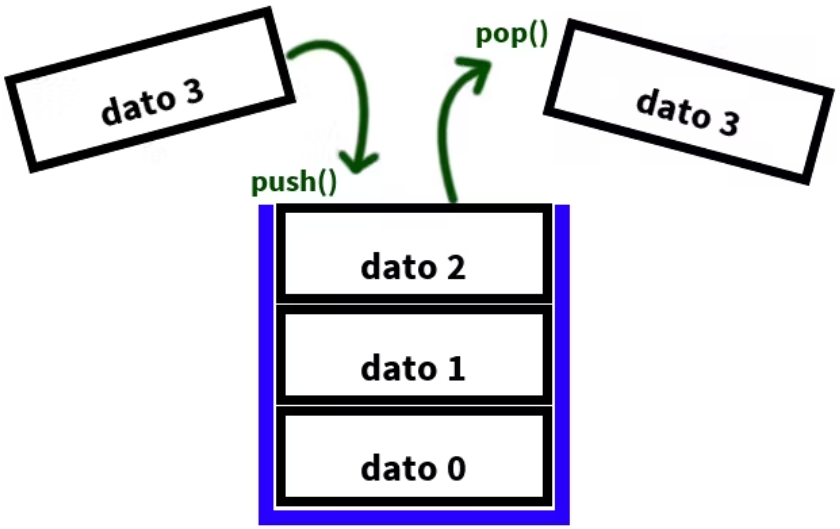


Introducción a las estructuras de datos

- Definición de estructuras de datos.
- Importancia en la programación y resolución de problemas.
- Comparación entre arrays y listas enlazadas.
- Operaciones básicas y complejidad de tiempo.

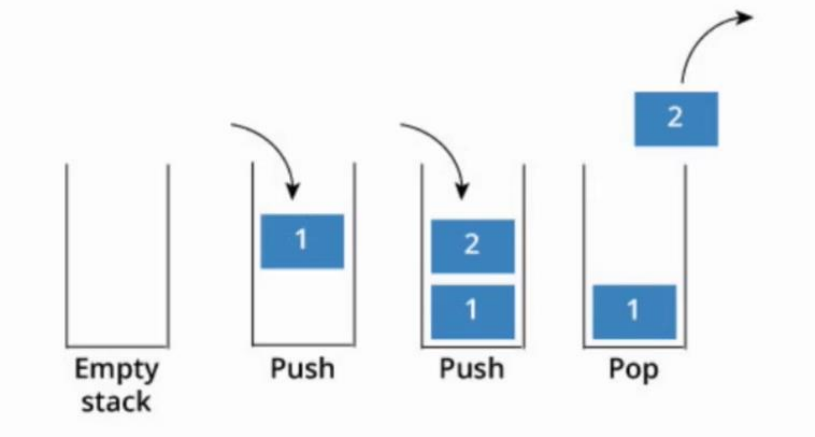
Introducción a las estructuras de datos

Es una forma particular de organizar información en un computador para que pueda ser utilizada de manera eficiente. Diferentes tipos de estructuras de datos son adecuados para diferentes tipos de aplicaciones, y algunos son altamente especializados para tareas específicas.



Introducción a las estructuras de datos

Las estructuras de datos son medios para manejar grandes cantidades de información de manera eficiente para usos tales como grandes bases de datos y servicios de indización de Internet. Por lo general, las estructuras de datos eficientes son clave para diseñar algoritmos eficientes. Algunos métodos formales de diseño de lenguajes de programación destacan las estructuras de datos, en lugar de los algoritmos, como el factor clave de organización en el diseño de software.

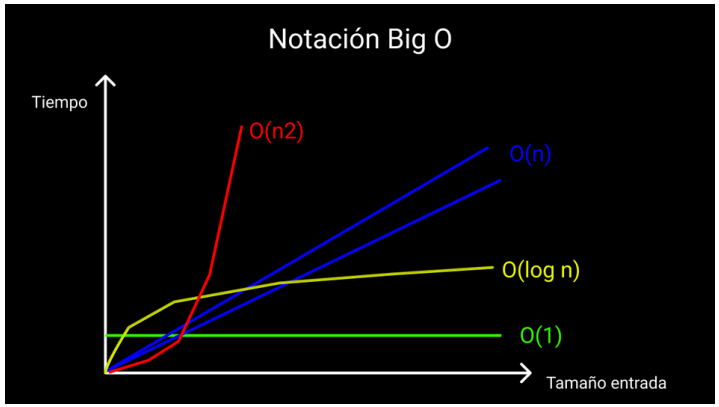


Introducción a las estructuras de datos

Más precisamente, una estructura de datos es una colección de valores, las relaciones entre ellos y las funciones y operaciones que se pueden aplicar a los datos, es decir, es una estructura algebraica de datos.

Importancia en la programación y resolución de problemas: Optimización de Espacio y Tiempo

- Las estructuras de datos permiten almacenar y acceder a los datos de manera eficiente, lo que ayuda a optimizar el uso de la memoria.
- Algoritmos y estructuras de datos bien diseñados pueden mejorar significativamente la velocidad de ejecución de un programa, lo que es crucial para aplicaciones en tiempo real o grandes conjuntos de datos.



Importancia en la programación y resolución de problemas: Eficiencia en Operaciones:

La elección adecuada de estructuras de datos puede mejorar la eficiencia de las operaciones básicas, como la búsqueda, inserción y eliminación de elementos. Por ejemplo, las listas enlazadas son eficientes para inserciones y eliminaciones en posiciones intermedias, mientras que los arrays son rápidos para el acceso directo a elementos.

Importancia en la programación y resolución de problemas: Resolución de Problemas Complejos

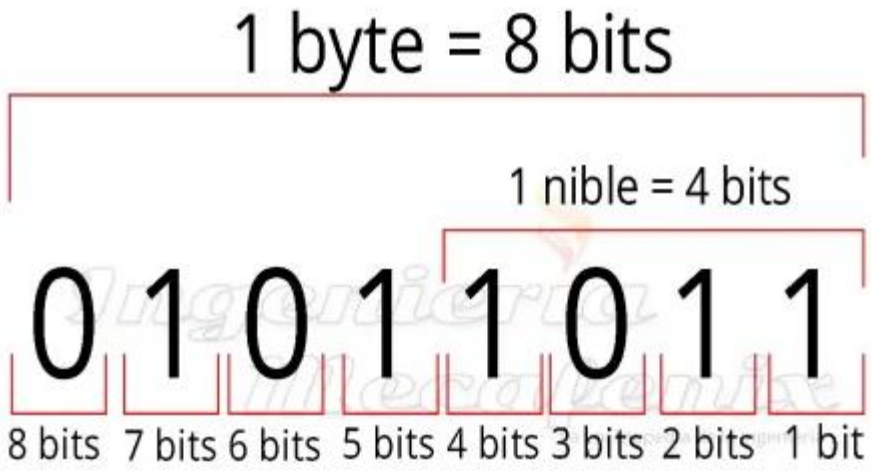
- Muchos problemas en ciencia de la computación y desarrollo de software se pueden modelar y resolver de manera más efectiva utilizando estructuras de datos adecuadas.
- Las estructuras de datos son esenciales para representar relaciones complejas entre datos, como redes sociales, mapas, grafos, etc.

Importancia en la programación y resolución de problemas: Facilita la Mantenibilidad del Código

- El uso de estructuras de datos claras y eficientes hace que el código sea más fácil de entender y mantener.
- Ayuda a evitar la redundancia de código y promueve prácticas de desarrollo limpias y estructuradas.

Concepto: tipo de datos

Un programa de computadora es un conjunto autocontenido de instrucciones para operar una computadora que produce un resultado especifico. Para realizar las instrucciones se utilizan los **datos**, los cuales son de diferentes tipos que para poder aplicarlos en la computadora se requiere que estos se representen de forma binaria.

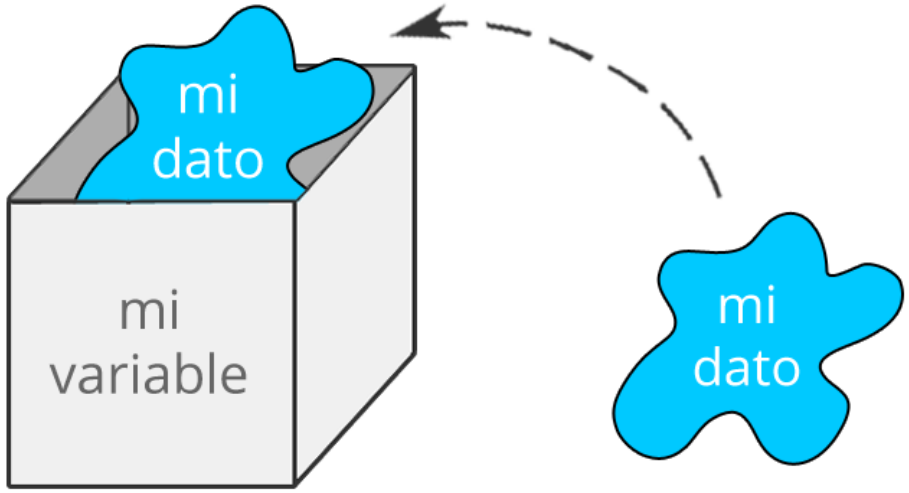


Concepto: tipo de datos

Hexadecimal	**2001:0DB8:0:AC10:0:FE01:0:0123**
Decimal	**8193:3512:0:44048:0:65025:0:291**
Binario	**1000000000000001:0000110110111000:0000000000000000:1010110000010000:0000000000000000:1111111000000001:0000000100100011**

Concepto: tipo de datos

Para evaluar la información por medio de un programa se usan variables las cuales se colocan al **principio de un bloque de un programa**, estos son los tipos de datos donde se declara una lista de variables que se marcan como tipos de datos, al marcar un tipo de dato esto hace que cada una de las variables tengan propiedades.



Concepto: tipo de datos

- ASCII (American Standard Code for Information Interchange) que se traduce al Español como Código Estándar Estadounidense para el Intercambio de Información, es un código de caracteres basado en el alfabeto latino, tal como se usa en ingles moderno y en otras lenguas occidentales.
- Fue creado en 1963 por el Comité Estadounidense de Estándares como una evolución de los conjuntos de códigos utilizados entonces en telegrafía.
- En 1967, se incluyeron las minúsculas, y se redefinieron algunos códigos de control para formar el código conocido como US-ASCII.
- El código ASCII utiliza 7 bits para representar los caracteres, aunque inicialmente empleaba un bit adicional (bit de paridad) que se usaba para detectar errores en la transmisión de los datos. A

Concepto: tipo de datos

Caracteres ASCII de control

00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sínc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles

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	_				

ASCII extendido (Página de código 437)

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

www.elCodigoAscii.com.ar

los más consultados

\	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)

de uso frecuente (idioma español)

ñ	alt + 164
Ñ	alt + 165
@	alt + 64
¿	alt + 168
?	alt + 63
¡	alt + 173
!	alt + 33
:	alt + 58
/	alt + 47
\	alt + 92

vocales con acento (acento agudo español)

á	alt + 160
é	alt + 130
í	alt + 161
ó	alt + 162
ú	alt + 163
Á	alt + 181
É	alt + 144
Í	alt + 214
Ó	alt + 224
Ú	alt + 233

vocales con diéresis

ä	alt + 132
ë	alt + 137
ï	alt + 139
ö	alt + 148
ü	alt + 129
Ä	alt + 142
Ë	alt + 211
Ï	alt + 216
Ö	alt + 153
Ü	alt + 154

símbolos matemáticos

½	alt + 171
¼	alt + 172
¾	alt + 243
¹	alt + 251
³	alt + 252
²	alt + 253
ƒ	alt + 159
±	alt + 241
×	alt + 158
÷	alt + 246

símbolos comerciales

$	alt + 36
£	alt + 156
¥	alt + 190
¢	alt + 189
¤	alt + 207
®	alt + 169
©	alt + 184
ª	alt + 166
º	alt + 167
°	alt + 248

comillas, llaves paréntesis

"	alt + 34
'	alt + 39
(	alt + 40
)	alt + 41
[	alt + 91
]	alt + 93
{	alt + 123
}	alt + 125
«	alt + 174
»	alt + 175

Representación binaria

La representación binaria de los datos se usa en el lenguaje de máquina, todos los tipos de datos tienen su representación binaria por lo que cada tipo de dato tiene un tamaño de espacio en la memoria de la computadora.

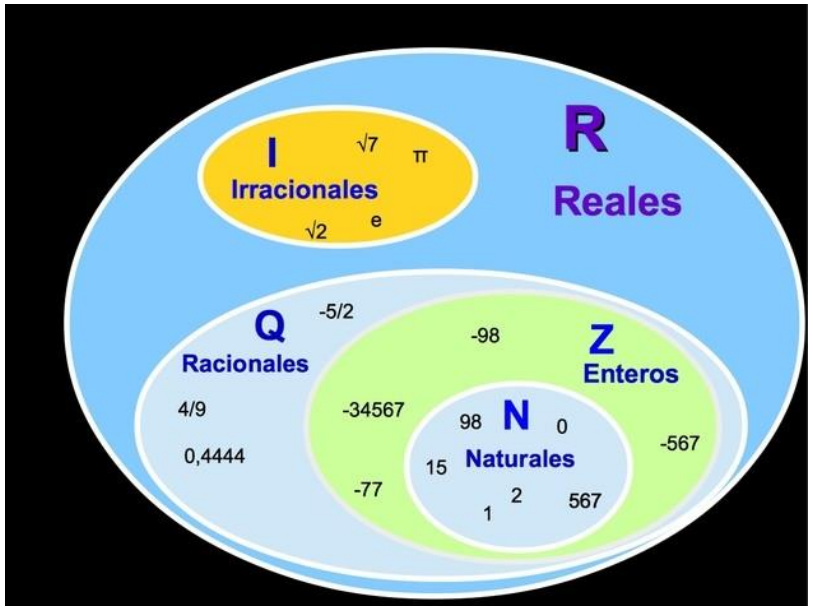
Los tipos de datos determinan **cuanto espacio de almacenamiento se debe permitir** junto con los tipos de operaciones permitidas.

8	4	2	1	⟹ Potencias de 2
2^3	2^2	2^1	2^0	
1	0	1	1_2	⟹ Número binario

Tipo de datos

Los tipos de datos se dividen en familias. Las cuales son:

- **La familia de enteros:** sirve para representar a los números enteros, que son los números naturales (positivos y negativos).
- **La familia de flotantes:** sirve para representar a los números reales que abarcan a los números irracionales y los racionales.



Tipo de datos

Carácter con signo	1 byte	-2^7 *a* $2^7 - 1$ (−128 a 127)
Carácter sin signo	1 byte	0 *a* 28 − 1 (0 a 255)
short	2 byte	-2^{15} *a* $2^{15} - 1$ (−32768 a 32767)
short sin signo	2 byte	0 *a* $2^{16} - 1$ (0 a 65535)
long entero	4 byte	2^{31} *a* $2^{31} - 1$ (2,147,483,648 a 2,147,483,647)
entero	2 o 4 byte	

Tipo de datos

```c
#include <stdbool.h>

int numero_entero = 10;

float numero_decimal = 3.14;
double otro_decimal = 2.71828;

char caracter = 'A';

char cadena[] = "Hola, mundo!";


bool es_verdadero = true;

int *puntero_entero;

typedef unsigned long int EnteroLargo;
EnteroLargo numero_grande = 9999999999;

```

```java
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        // Your code here!
        int numeroEntero = 10;
        float numeroDecimal = 3.14f; // La "f" indica que es un float en Java
        double otroDecimal = 2.71828;
        char caracter = 'A';
        String cadena = "Hola, mundo!";
        boolean esVerdadero = true;
    }
}
```

INSTITUCIÓN UNIVERSITARIA DE ENVIGADO

Ciencia , educación y desarrollo

Vigilada Mineducación

Preguntas rápidas

1. ¿Cuál es el tipo de dato?
   a) 4
   b) 23.14
   c) "Hola, mundo“
   d) True
   e) False
   f) "123“

2. Convierte los siguientes valores al tipo de dato especificado.
a) De entero a cadena: 123
b) De cadena a entero: "456“
c) De decimal a entero: 7.89
d) De booleano a cadena: False

Preguntas rápidas

```c
#include <stdio.h>

int main() {
    int num = 123;
    char str[10];
    sprintf(str, "%d", num);
    printf("La cadena es: %s\n", str);
    return 0;
}
```

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

int main() {
    char str[] = "456";
    int num = atoi(str);
    printf("El número entero es: %d\n", num);
    return 0;
}
```

```c
#include <stdio.h>

int main() {
    double num = 7.89;
    int integerPart = (int)num;
    printf("La parte entera es: %d\n", integerPart);
    return 0;
}
```

Preguntas rápidas

3. Realiza las siguientes operaciones y determina el tipo de dato del resultado.
    a) 7 + 5
    b) 10 - 3.5
    c) "Hola" + " " + "mundo“
    d) 3 * 4

Tipos en lenguajes de programación

Clases de lenguajes de programación según la tipificación:

- Lenguaje fuertemente tipificado: todos los datos deben tener un tipo declarado.
- Lenguajes no tipificados: no se requiere que los datos tengan un tipo declarado explícitamente.

Clasificación de tipos de datos

- **Estáticos:** el tamaño que ocupa una variable en memoria no puede cambiar en tiempo de ejecución
  - Simple: tipos básicos ofrecidos en general por los lenguajes de programación
    - Ordinales: el conjunto de valores que representa se puede contar (entero, lógico, carácter)
    - No ordinales: el conjunto de valores que representa no se puede contar (reales)
  - Estructurado:
    - Colecciones ordenadas (arreglos) y no ordenadas (conjuntos) de elementos de un mismo tipo
    - Estructuras conteniendo elementos de tipos diferentes

Dinámicos: el tamaño de que ocupa una variable en memoria puede ser modificado en tiempo de ejecución (apuntadores)

Abstracción de datos

Se define como la separación de las propiedades lógicas de los tipos de datos de su implementación. Los datos son los **sustantivos de un programa**. En
el mundo real los objetos son manipulados, en un programa de computadora
la información es procesada.

Abstracción de datos

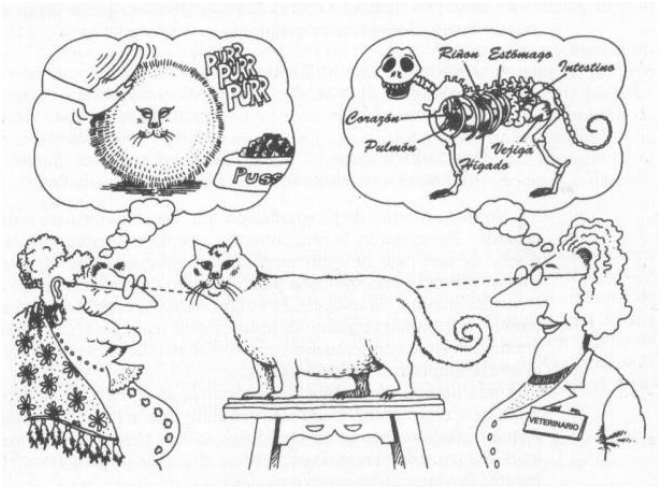
Cuando se habla de un entero es fácil usarlo pues lo hemos convertido en una generalidad ya que todo el tiempo lo aplicamos para realizar programas. Y un entero es una abstracción de un tipo de dato. Como ejemplo la siguiente operación es una abstracción de datos:

resultado = a + b;

Puesto que usa un modelo de datos, donde solamente se utilizan las características más relevantes. Este modelo es ideal para realizar operaciones utilizando un programa de computadora donde se aplica un lenguaje.

Conceptos fundamentales

Los datos del mundo real son muchos y la mayora son innecesarios, la limitante de la computadora es que no puede considerar todos los datos que representan a un objeto del mundo real, sin embargo, se hacen consideraciones de tal manera que con el menor número de datos se pueda representar en la computadora el objeto del mundo real, estas consideraciones tienen por nombre abstracción de los datos.



Conceptos fundamentales

La abstracción de los datos es la consideración de las características primarias, estas características se pueden representar por medio de conjuntos de datos, los datos adecuados. La forma de seleccionar los tipos de datos se hace siguiendo dos pasos. El primero es seleccionar el tipo de dato que representa la abstracción de la realidad adecuadamente, el segundo es la representación de la información. Los tipos de datos representan características como en matemáticas los números reales, complejos, o variables lógicas.

Abstracción de datos: funciones

Una función es una herramienta de abstracción para separar su comportamiento físico de su implementación. La separación es importante por dos razones: Es fácil de diseñar programas usando descomposición funcional (descomponer un gran problema en pequeños subproblemas, cada uno expresado como una llamada a función).

```
1 def suma(a, b):
2     resultado = a + b
3     return resultado
```

Abstracción de datos: funciones

Las funciones caen en dos categorías:

- Aquellas que regresan **un solo valor de función**
- Aquellas que no (procedimientos)

Al diseñar e implementar funciones se requiere poner atención cuidadosa al flujo de datos de la lista de parámetros. El flujo de datos es el flujo de la información del llamador hacia la función y de la función hacia el llamador.

```
float volumen(int x, float y, float z);
```

Abstracción de datos: funciones

```
float volumen(int x, float y, float z);

   float volumen(int a, float b, float c)
   {
   float v = a*b*c;
   }

 float cubo1 = volumen(b1, w1, h1);
```

Ejemplo

```c
#include <stdio.h>

int suma(int a, int b) {
    return a + b;
}

int main() {
    int resultado = suma(5, 7);
    printf("La suma es: %d\n", resultado);
    return 0;
}
```

Ejemplo

```c
#include <stdio.h>

int cuadrado(int num) {
    return num * num;
}

int main() {
    int resultado = cuadrado(4);
    printf("El cuadrado es: %d\n", resultado);
    return 0;
}
```

Ejemplo

```c
#include <stdio.h>

float sumaFloat(float a, float b) {
    return a + b;
}


int cuadradoInt(int num) {
    return num * num;
}

char concatenarChars(char c1, char c2) {
    char resultado[3];
    resultado[0] = c1;
    resultado[1] = c2;
    resultado[2] = '\0';
    return resultado[0];
}

int main() {

    float resultadoFloat = sumaFloat(3.14, 2.5);
    int resultadoInt = cuadradoInt(4);
    char resultadoChar = concatenarChars('A', 'B');


    printf("Resultado de la suma de flotantes: %.2f\n", resultadoFloat);
    printf("Resultado del cuadrado de un entero: %d\n", resultadoInt);
    printf("Resultado de la concatenación de caracteres: %c\n", resultadoChar);

    return 0;
}
```