Estructura de datos

MODULO 1. Representación de datos

1.1 Tipos de Datos Primitivos

1.2 Tipos de datos estructurados

1.2.1 Arreglos unidimensionales, bidimensionales y cadenas de caracteres

1.2.2 Registros o Estructuras (unión y estructura)

1.3 Definición de estructura de datos

Representación de Datos

Tipos de datos primitivos

Tipos de Datos Estructurados

Tipos de Datos Simples o Primitivos

Dato	Longitud	Rango
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	3.4 * (10**-38) to 3.4 * (10**+38)
double	64 bits	1.7 * (10**-308) to 1.7 * (10**+308)
long double	80 bits	3.4 * (10**-4932) to 1.1 * (10**+4932)

Tipos de Datos Simples o Primitivos

- Surgen de la necesidad de tener una clasificación específica de la información en un programa de computadora.
- Estos cuentan con una longitud definida y un rango limitado de valores.
- Consisten de un conjunto posible de valores y un conjunto permitido de operaciones aplicables a dichos valores.

Datos

- Son valores básicos que se manipulan en un programa y sobre los que se realizan operaciones. Estos se almacenan en posiciones o localidades de la memoria física disponible de la computadora.

- **Variable:** dato cuyo valor puede variar. La declaración de variables supone:

  - Darle un nombre o identificador.
  - Asignarle un tipo de dato.

- **Declaración de variables:**

  - **<tipo_de_dato> variable;**
  - **<tipo_de_dato> variable1, variabler2, … , variableN;**

Datos

- Ejemplos de declaración de variables:

  - int x; //variable entera x de tipo int
  - double y; //variable real y de tipo double
  - char c; //variable c de tipo char
  - int i, j, k; //múltiples declaraciones en una sóla línea
  - int a = 0; //variable a de tipo int inicializada a 0

Datos

- Cuando declaramos una variable, definimos de forma implícita:
  - La longitud del dato (número de bits o bytes).
  - Operaciones que se pueden realizar con ese dato.
  - Conjunto de valores posibles.

Datos

- Consideraciones para el uso de variables:
  - Cuando una variable se declara, el compilador le reserva espacio en la memoria, pero no modifica la información que había en dicho espacio.
  - Toda variable debe ser inicializada antes de usar su valor en el programa.
  - Las variables pueden ser inicializadas a un valor concreto cuando son declaradas.
  - Para ello, en la declaración se añade el signo de igual más una constante.

Datos

- **Constante**: dato cuyo valor no puede variar.
  - Al igual que las variables, las constantes guardan datos, pero su valor no varía. Existen tantos tipos de constantes como tipos de datos. Podemos expresar una constante según el tipo de dato. Las constantes pueden aparecer con su valor directamente o mediante un identificador.
- **Constantes numéricas:** Son valores numéricos (enteros o reales). Se permiten también constantes octales y hexadecimales.
  - **const tipo_de_dato nombre = valor;**
- **Constantes simbólicas**: a las cuales se les asocia un nombre o identificador. Se declaran como:
  - **#define nombre_constante valor**

Datos

- Ejemplos de declaración de constantes

  - **#define M 10**
  - **#define N 10.0 /* Es un double */**
  - **#define PI 3.14159265 /* double */**
  - **#define PIf 3.14159265f /* float */**
  - **#define A -1.2345e-17 /* double en notación científica */**
  - **#define salto_de_linea '\n'**
  - **#define HOLA "HOLA“**
  - **const int LIMITE = 30;**
  - **const float ALTURA = 2.5f;**

Datos

- **Tipos de datos definidos**: permiten dar nombres a tipos de datos que ya existen, siendo estos más acordes con aquello que representan.

- Sintaxis:

  - **typedef tipo_basico nombre;**

- Declaración:

  - typedef float Kg;
  - typedef float Mts;

Datos

- Uso al declarar las variables:
  - Kg peso;
  - Mts longitud;

Operadores

- Los operadores, junto con los operandos, forman expresiones. En una expresión, los operandos pueden ser constantes, variables o llamadas a funciones que devuelvan valores. Acorde al tipo de operación que realizan, se clasifican en:

  - Aritméticos
  - Relacionales
  - Lógicos
  - Para el tratamiento de bits
  - Especiales

Operadores

Operador	Uso	Descripción
+	*op1 + op2*	Suma op1 y op2
-	*op1 - op2*	Resta op2 de op1
*	*op1 * op2*	Multiplica op1 por op2
/	*op1 / op2*	Divide op1 por op2
%	*op1 % op2*	Calcula el resto de dividir op1 entre op2

Operadores aritméticos binarios

Operadores

Operador	Uso	Descripción
++	*op*++	Incrementa op en 1; se evalúa al valor anterior al incremento
++	++*op*	Incrementa op en 1; se evalúa al valor posterior al incremento
--	*op*--	Decrementa op en 1; se evalúa al valor anterior al incremento
--	--*op*	Decrementa op en 1; se evalúa al valor posterior al incremento

Operaciones con ++ y --

Operadores

Operador	Uso	Devuelve verdadero si
>	op1 > op2	op1 es mayor que op2
>=	op1 >= op2	op1 es mayor o igual que op2
<	op1 < op2	op1 es menor que op2
<=	op1 <= op2	op1 es menor o igual que op2
==	op1 == op2	op1 y op2 son iguales
!=	op1 != op2	op1 y op2 son distintos

Operadores relacionales o de comparación

Operadores

Operador	Uso	Devuelve verdadero si
&&	*op1 && op2*	op1 y op2 son ambos verdaderos, condicionalmente evalúa op2
&	*op1 & op2*	op1 y op2 son ambos verdaderos, siempre evalúa op1 y op2
||	*op1 || op2*	op1 o op2 son verdaderos, condicionalmente evalúa op2
|	*op1 | op2*	op1 o op2 son verdaderos, siempre evalúa op1 y op2
!	*!op*	op es falso

Operadores lógicos o condicionales

Operadores

Operador	Uso	Equivalente a
+=	*op1 += op2*	op1 = op1 + op2
-=	*op1 -= op2*	op1 = op1 - op2
*=	*op1 *= op2*	op1 = op1 * op2
/=	*op1 /= op2*	op1 = op1 / op2
%=	*op1 %= op2*	op1 = op1 % op2
&=	*op1 &= op2*	op1 = op1 & op2

Operadores de atajo de asignación

Estructura de un Programa en C

```
/*
Comentarios inciales del programador
en relación a la descripción del programa
Fecha: fecha de creación del programa
Autor(es): Autor(es) del programa
*/
#include ____    //inclusión de bibliotecas de funciones
#define ____      //definición de valores constantes y macros
[typedef ____ ]   //definición de nuevos tipos de datos
[Definición de variables globales] //variables y sus inicializaciones
[Prototipos de funciones]
int main(void) //función principal del programa
{
[declaración de variables] //variables y sus inicializaciones
[instrucciones del programa] //operaciones y llamadas a funciones
return 0; //valor de retorno de la función main
}
```

Estructura de un Programa en C

- Un programa algo más complicado es el siguiente:

```
//Nuestro primer programa escrito en lenguaje C
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   printf("Hola mundo!!!\n\n");
   system("Cls");
   return 0;
}
```

Estructura de un Programa en C

- Un programa se compone de:
  - **Estructuras de datos:** organizan los datos de un programa.
  - **Operaciones primitivas elementales:** acciones que se ejecutan sobre los datos para transformarlos en información.
  - **Estructuras de control:** métodos que existen para dirigir el flujo de acciones que la computadora deberá ejecutar sobre los datos manejados por el programa.