LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

INGENIERÍA EN INFORMÁTICA

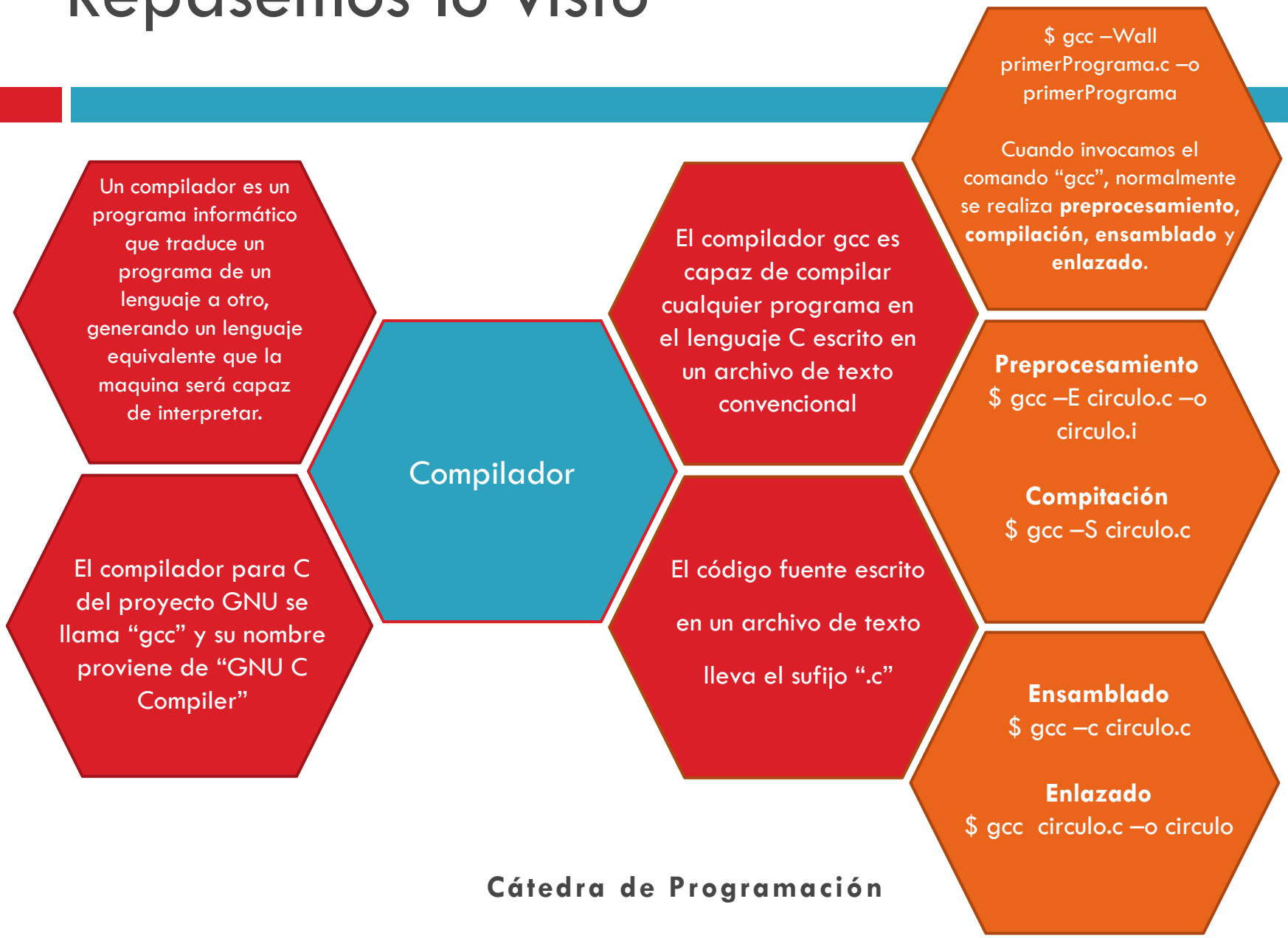
PROGRAMACIÓN

Unidad N° 3 – Parte 1: Concepto de Tipos de Datos. Tipos de Datos Simples. Constantes y variables. Entrada y Salida básicas.

2022

Lic. Mariela A. Velázquez

Repasemos lo visto



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

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.
- Por 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 Simbólicas

- Una manera de evitar el uso de “números mágicos”, que aparecen en nuestro código sin ninguna explicación es dándoles nombres significativos.
- Una línea **#define** define una constante simbólica. Por ejemplo:

#define maximo 200

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.

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	\0
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: cadena de control que indica, por regla general, los modificadores que harán referencia al tipo de dato de los argumentos.

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.

Función de Entrada: scanf

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);
```

```c
#include <stdio.h>

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

    return 0;
}
```