Principios de Programación

Introducción a la Programación

1. Conceptos de Computación

1.1. Qué es una computadora

Una computadora es un dispositivo capaz de ejecutar cálculos y tomar decisiones lógicas a velocidades millones y miles de millones de veces más rápido de lo que pueden hacerlo los seres humanos.

Las computadoras procesan datos bajo el control de un conjunto de instrucciones que se conocen como *programas de computación*. Estos programas guían a la computadora a través de conjuntos ordenados de acciones especificados por personas a las que se conoce como *programadores*.

Los varios dispositivos (teclado, pantalla, discos, memoria, unidades procesadoras) se conocen como *hardware*. Los programas de computación se conocen como el *software*.

1.2. Lenguajes de programación

Los programadores escriben instrucciones en diferentes lenguajes de programación, algunos comprensibles en forma directa por la computadora y otros que requieren pasos intermedios de traducción.

Existen hoy día cientos de lenguajes de computadora. Estos pueden ser categorizados en tres tipos generales:

1. Lenguajes de máquina
2. Lenguajes ensambladores
3. Lenguajes de alto nivel

Cualquier computadora solo puede entender directamente su propio *lenguaje de máquina*. Este está relacionado intimamente con el diseño del hardware de la computadora. Estos lenguajes son dependientes de la máquina. En general consisten de cadenas de números que instruyen a las computadoras para que ejecuten sus operaciones más elementales una a la vez.

ejemplo:

1300042774
1400593419
1200274027

Conforme las computadoras se hicieron más populares se hizo aparente que la programación en lenguaje de máquina era lenta y tediosa para los programadores. Se empezaron entonces a utilizar abreviaturas similares al ingles para representar las operaciones elementales de la computadora. Estas abreviaturas formaron la base de los *lenguajes ensambladores*. Se desarrollaron *programas de traducción* para convertir los programas del lenguaje ensamblador a lenguaje de máquina.

ejemplo:

```
LOAD BASE
ADD VALOR
STORE RESULTADO
```

La utilización de las computadoras aumentó con rapidez con la llegada de los lenguajes ensambladores pero aún con estos se necesitaban muchas instrucciones para llevar a cabo inclusive tareas sencillas. Para acelerar el proceso de programación se desarrollaron *lenguajes de alto nivel* en los cuales se pueden escribir sentencias simples que llevan a cabo tareas sustanciales.

2. Los fundamentos del entorno de C

Todos los sistemas C consisten en general de tres partes: el entorno, el lenguaje y la biblioteca estándar.

Veamos el entorno típico de desarrollo de C:

1. Fase 1 : Editor. El programa es creado en el editor y almacenado en disco.
2. Fase 2 : Preprocesador. El programa preprocesador procesa el código. Se ejecuta antes de la traducción. Se encarga de la inclusión de otros archivos en el archivo a compilar y en el reemplazo de símbolos especiales con texto de programa. Es invocado por el compilador antes de la traducción del programa a lenguaje de máquina.
3. Fase 3 : Compilador. El compilador crea el código objeto y lo almacena en disco.
4. Fase 4 : Enlazador. El enlazador vincula el código objeto con las bibliotecas, crea el archivo a.out y lo almacena en disco.
5. Fase 5 : Cargador. El cargador coloca el programa en memoria. Antes de que un programa pueda ser ejecutado debe ser colocado en memoria.
6. Fase 6 : CPU. La CPU toma cada una de las instrucciones y las ejecuta almacenando posiblemente nuevos valores de datos conforme se ejecuta el programa.

Veremos el lenguaje a lo largo del curso.

Veamos la biblioteca estandar.

Los programas C consisten de módulos que se denominan funciones. En general, además de las funciones que uno programa se utilizan funciones ya existentes que se encuentran en la biblioteca estandar de C.

Utilizar funciones ya existentes se conoce como reutilización de software.

En general utilizaremos:

- Funciones de la biblioteca estandar C
- Funciones que crea uno mismo
- Funciones creadas por otros programadores

3. Algoritmos

Antes de escribir un programa para resolver un problema es esencial tener comprensión completa del mismo y un método planeado de forma cuidadosa para su resolución.

La solución a cualquier problema de cómputo involucra la ejecución de una serie de acciones en un orden especifico. Un *algoritmo* es un procedimiento que resuelve un problema en términos de:

1. las acciones a ejecutarse
2. el orden en el cual estas deben ejecutarse

Veamos un ejemplo, el algoritmo de "levantarse"que debe seguir un ejecutivo para salir de la cama y llegar al trabajo:

Salir de la cama
Quitarse los pijamas
Darse una ducha
Vestirse
Desayunar
Utilizar el vehiculo para llegar al trabajo

Si cambiaramos el orden y colocaramos Vestirse y Darse una ducha intercambiados el ejecutivo llegaría al trabajo mojado.

4. Pseudocódigo

El pseudocódigo es un lenguaje artificial e informal que auxilia a los programadores a desarrollar los algoritmos. Este es similar al lenguaje natural, es amigable aunque no se trate de un lenguaje verdadero de programación de computadoras.

Es un lenguaje intermedio entre el lenguaje natural y los lenguajes de programación.

Los programas en pseudocódigo no son ejecutables, se utilizan porque ayudan al programador a pensar un programa antes de intentar escribirlo en un lenguaje de programación como C.

Un programa preparado cuidadosamente en pseudocódigo puede ser convertido con facilidad en el correspondiente programa en C. Esto se lleva a cabo en muchos casos reemplazando enunciados en pseudocódigo por sus equivalentes en C.

A continuación veremos las ïnstrucciones"que utilizaremos para escribir pseudocódigo:

- Definir constante con valor x
- Definir variable con valor x
- Realizar operaciones aritméticas (escribimos la operación)
- Leer datos
- Imprimir datos
- Si se cumple condicion entonces realizo sentencia
- Si se cumple condicion entonces realizo sentencia1 si no realizo sentencia2
- Mientras se cumpla condicion ejecutar sentencia

5. Diagramas de flujo

Por lo general en un programa los enunciados son ejecutados uno despues de otro, en el orden en que aparecen escritos. Esto se conoce como *ejecución secuencial*.

Varios enunciados de C, que pronto analizaremos le permiten al programador especificar que el enunciado siguiente a ejecutar pueda ser otro diferente del que sigue a continuación.

Un diagrama de flujo es una representación gráfica de un algoritmo. Los diagramas de flujo se trazan utilizando símbolos de uso especial como ser rectángulos y rombos. Estos símbolos están conectados entre si por flechas conocidas como *líneas de flujo*.

Al igual que el pseudocódigo, los diagramas de flujo son útiles para el desarrollo y la representación de algoritmos.

El diagrama de flujo correspondiente a una acción como ser un cálculo o una operación de entrada/salida es el rectángulo.

El diagrama correspondiente a una secuencia es una secuencia de rectangulos. Por ejemplo, el diagrama correspondiente a:

sumar grado a total
sumar 1 a contador

es el de la figura 1.

Cuando dibujamos una porción de un algoritmo colocamos pequeños círculos conocidos como símbolos de conexión.

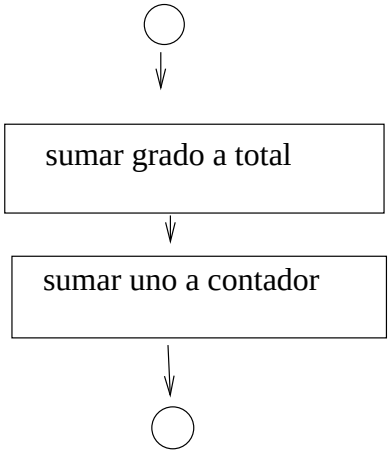

Figura 1: Diagrama de flujo de secuencia

Quiza el simbolo de diagrama de flujo más importante es el rombo, también conocido como *símbolo de decisión*, que indica donde se debe tomar una decisión.

Supongamos tenemos el siguiente pseudocódigo:

Si se cumple grado es mayor o igual a 60 entonces
Imprimir “Aprobado”

el diagrama correspondiente es el de la figura 2.

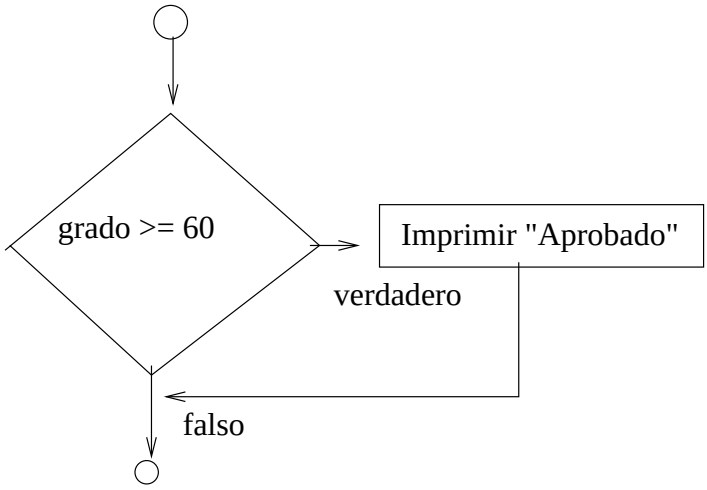

Figura 2: Diagrama de flujo de selección

Consideremos ahora el Si-si no. Este permite especificar que se ejecuten acciones distintas cuando la condición sea verdadera y cuando sea falsa. Dado el siguiente pseudocódigo,

Si se cumple grado es mayor o igual a 60 entonces
   Imprimir “Aprobado”
si no
   Imprimir “Aplazado”

el diagrama correspondiente es el de la figura 3.

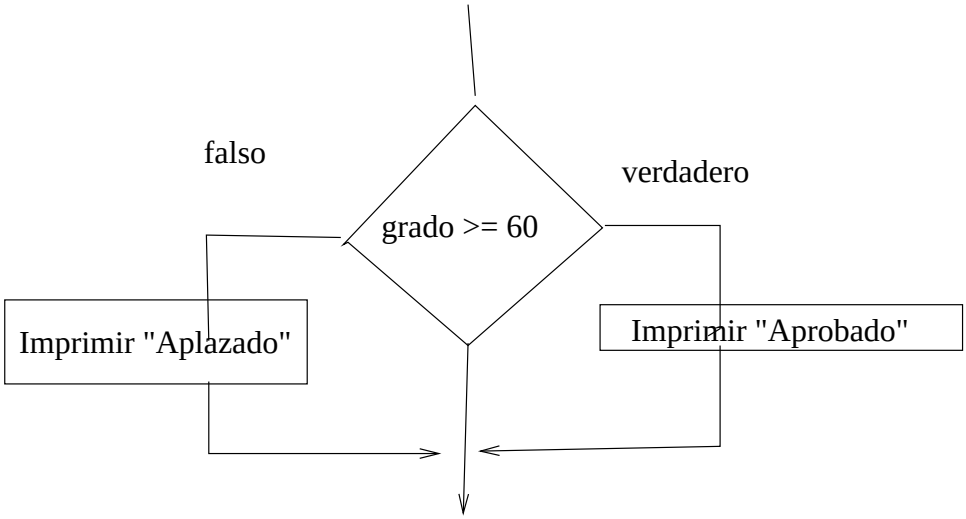

Figura 3: Diagrama de flujo de doble selección

Consideremos ahora el ”Mientras”. Supongamos queremos calcular la primera potencia de 2 superior a 1000. El pseudocódigo es:

Definir producto=2.
Mientras producto <= 1000
        multiplicar producto por 2.
Imprimir producto

el diagrama correspondiente es el de la figura 4.

6. Casos de Estudio (pseudocódigo)

6.1. Repetición controlada por contador

Consideremos el siguiente enunciado:

Una clase de diez alumnos hizo un examen. Las calificaciones (enteros en el rango de 0 a 100) correspondientes a este examen están a su disposición. Determine el promedio de la clase en este examen.

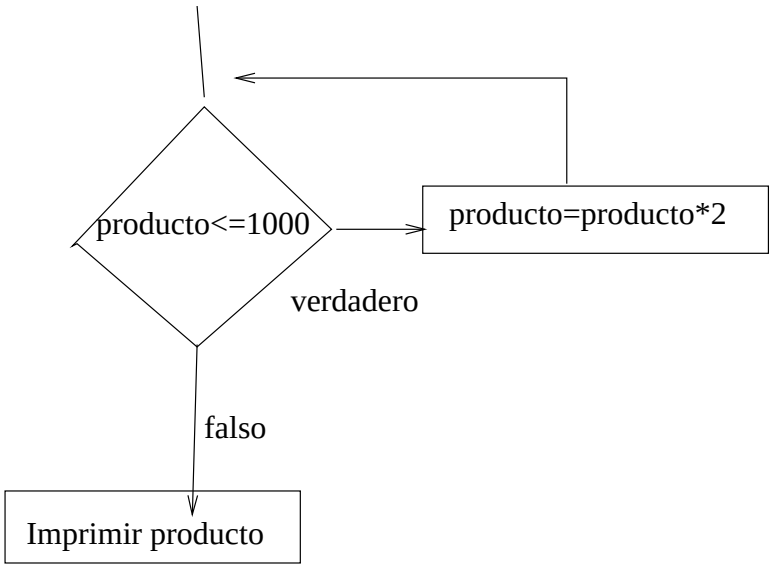

Figura 4: Diagrama de flujo de mientras

El promedio de la clase es igual a la suma de las calificaciones dividida por el número de alumnos. Para resolver el problema debemos leer las calificaciones, calcular el promedio e imprimir el resultado.

El contador que utilizaremos es la cantidad de alumnos, mientras no pasemos de los 10 alumnos leeremos las calificaciones.

Utilizaremos además una variable total en la que sumaremos las calificaciones. Esta variable debe inicializarse en 0.

El pseudocódigo es el siguiente:

```
Inicializar total en 0
Inicializar nro_alumno a 1
Mientras nro_alumno menor o igual a 10
        Leer siguiente calificacion
        Sumar calificacion a total
        Sumar 1 a nro_alumno
Calcular promedio = total /10
Imprimir promedio
```

6.2. Repetición controlada por centinela

El problema es igual al anterior salvo que en vez de considerar 10 estudiantes consideramos una cantidad arbitraria.

Ingresaremos datos y cuando hayamos terminado con todos los datos ingresaremos -1.

El pseudocódigo es el siguiente:

```
Inicializar total en 0
Inicializar cant_alumnos a 0
```

Leer centinela
Mientras centinela distinto de -1
Leer siguiente calificacion
Sumar calificacion a total
Sumar 1 a cant_alumnos
Leer centinela
Si cant_alumnos es distinto de 0 entonces
calcular promedio = total /cant_alumnos
Imprimir promedio
si no Imprimir "no se ingresaron calificaciones"

6.3. Estructuras de control anidadas

Consideremos el siguiente problema:

Una universidad ofrece un curso que prepara alumnos para un examen. La universidad desea saber que tan bien salieron sus alumnos en el examen. Se ingresará un 1 si el alumno pasó el examen y un 2 si lo reprobó. Se quiere saber total de aprobados y total de reprobados en un total de 10 alumnos.

El pseudocódigo es el siguiente:

Inicializar total en 0
Inicializar aprobados en 0
Inicializar reprobados en 0
Inicializar cant_alumnos a 1
Mientras cant_alumnos menor o igual a 10
Leer siguiente calificacion
Si calificacion = 1 sumar 1 a aprobados
si no sumar 1 a reprobados
Sumar 1 a cant_alumnos
Imprimir "Total de aprobados="
Imprimir aprobados
Imprimir "Total de reprobados="
Imprimir reprobados

7. Programas ejemplo

7.1. Primer Programa en C

Imprimiremos la linea de texto: "Bienvenido a C!"

Pseudocodigo
Imprimir "Bienvenido a C!"

Programa
/* Primer programa en C */

```
main ()
{
      printf("Bienvenido a C!\n");
      system("PAUSE");
}
```

Veamos las lineas del programa:

```
/* Primer programa en C */
```

es un comentario. Escribimos comentarios colocando /* al principio y */ al final. Colocamos comentarios para facilitar el entendimiento del programa. Cuando se ejecuta el programa los comentarios se ignoran estos son ignorados por el compilador C y no generan código. Los comentarios ayudan a otras personas a leer y comprender el programa.

La linea

```
main ()
```

forma parte de todo programa de C. Los paréntesis despues de main indican que es una función. Los programas en C contienen una o más funciones una de las cuales deberá ser main. Todos los programas en C empiezan a ejecutarse en main.

La llave izquierda { debe de iniciar el cuerpo de cada función. Una llave derecha correspondiente debe de dar por terminada cada función.

La linea:

```
printf("Bienvenido a C!\n");
```

instruye a la computadora para que ejecute una acción, en este caso que imprima en la pantalla la cadena de caracteres entre comillas.

printf es una instrucción, toda instrucción debe terminar con un punto y coma.

La instrucción printf es una instrucción de biblioteca no forma parte del lenguaje. Cuando se compila y enlaza un programa se localizan las funciones de la biblioteca y se insertan las llamadas a estas funciones dentro del programa objeto.

Veamos aspectos de printf:

Cada llamada a printf contiene una cadena de control de formato que describe el formato de la salida. En este ejemplo en particular la cadena consiste en el string (secuencia de caracteres) "Bienvenido a C! \n". El \n final indica que se realize un salto de linea es decir despues de imprimir "Bienvenido a C!"se salta al comienzo de la linea siguiente.

Por último la instrucción:

```
system("PAUSE");
```

indica que el programa desplegue el mensaje:

Presione una tecla para continuar . . .

y que espere la entrada de una tecla cualquiera.
Para utilizar la función system uno debe incluir la biblioteca iostream.h.

7.2. Segundo Programa en C

Imprimiremos un valor aproximado de PI.

Pseudocodigo
Definir constante con valor de PI
Imprimir constante

Programa

```
/* Imprime valor aproximado de PI en la pantalla */

#include <stdio.h>
#include <iostream.h>

#define PI 3.1415926

main()
{
    printf("El valor aproximado de PI es: %f\n", PI);
    system("PAUSE");
}
```

Utilizamos printf y system, luego incluimos las bibliotecas correspondientes mediante las directrices:

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

La directiva :

```
#define PI 3.1415926
```

define una constante simbólica PI cuyo valor es 3.1415926. Una constante simbólica es un identificador que se reemplaza con texto de reemplazo en el preprocesador de C antes de que el programa sea compilado.

Observar que PI es un número real. En este caso definimos una constante con su valor.

Veamos la instrucción **printf**:

la cadena de control de formato contiene texto "El valor aproximado de PI es", contiene %f y \n.

El texto se imprime tal cual se colocó en la instrucción. %f es un especificador de conversión. Indica que se va a imprimir un número en punto flotante. \n indica que luego de imprimir el número se avanze a la linea siguiente.

El número en punto flotante se toma de la constante PI.

Los valores impresos con %f siempre imprimen por lo menos un digito a la izquierda del punto decimal. Además se colocan 6 digitos de presición a la derecha del punto decimal.

En este ejemplo se imprime:

El valor aproximado de PI es: 3.141693

Cuando imprimimos números en punto flotante podemos utilizar otros especificadores de conversión, en particular %e, %E (para imprimir float y double). %e imprime la e en miniscula. %E imprime la E mayuscula. Igual que %f imprimen un dígito a la izquierda del punto decimal. %e y %E muestran los números en notación exponencial.

7.3. Tercer Programa en C

Leeremos valores enteros ingresados en el teclado, calcularemos la suma e imprimiremos el resultado.

Pseudocodigo
Leer valor de x
Leer valor de y
Calcular suma=x+y
Imprimir suma

Programa

```
/* Suma dos numeros enteros */
#include <stdio.h>
#include <iostream.h>

main()
{
      int x,y,suma;
      printf("Ingrese 1er entero\n");
      scanf(" %d",&x);
      printf("Ingrese 2do entero\n");
      scanf(" %d",&y);
      suma=x+y;
      printf("La suma es %d\n",suma);
      system("PAUSE");
}
```

La linea:

int x,y,suma;

es una declaración. x,y y suma son nombres de variables. int es un tipo de datos.

Las variables son posiciones de memoria en las que se almacenan valores. Se están declarando tres variables que contendrán valores enteros.

Todas las variables deben de declararse con un nombre y un tipo de datos. En este caso declaramos las tres variables juntas, podriamos haber puesto:

```
int x;
int y;
int suma;
```

Observar que las declaraciones terminan en ;. Las declaraciones de variables se pueden realizar en cualquier punto de un programa (debe ser antes de utilizarlas).

Los nombres de variable deben ser identificadores válidos. Un identificador es una serie de caracteres formados de letras, digitos y subrayados (_), que no se inicien con un digito.

Mayusculas y minusculas se consideran diferentes.

Veamos la primer instrucción **scanf**:

```
scanf(" %d",&x);
```

toma la entrada del teclado. Tiene como argumentos %d y &x. El primer argumento es la cadena de control que indica de que tipo de datos es el valor que debe ingresar el usuario. %d indica debe ser un entero. El segundo argumento indica en que variable se debe almacenar el número leido, en este caso x.

Debe colocarse & antes de la variable excepto cuando se está leyendo una cadena.

Podriamos leer x e y juntos con la siguiente instrucción: (en general podemos leer cualquier cantidad de variables)

```
scanf(" %d %d",&x, &y);
```

se deben ingresar la x y la y separados por espacio, nueva linea o tabulador.

La instrucción:

```
suma=x+y;
```

calcula la suma de x e y y coloca el resultado en la variable suma.

La instrucción printf

```
printf("La suma es %d\n",suma);
```

declara la suma a imprimir es un entero (con %d). Podriamos haber realizado la suma en la instrucción printf escribiendola del siguiente modo:

```
printf("La suma es %d\n",x+y);
```

en cuyo caso no es necesario declarar la variable suma.