TEMA 9
ESTRUCTURAS DE CONTROL. SENTENCIAS

9.1. Estructura Secuencial

9.2. Estructuras de Selección

9.3. Estructuras de Repetición

9.4. Entrada/Salida por Consola

9.1. Estructura Secuencial

Mediante la programación estructurada se ilustra que cualquier programa puede construirse utilizando tres tipos de estructuras:

- Estructuras Secuenciales
- Estructuras alternativas o condicionales o de selección o de control de flujo de ejecución
- Estructuras repetitivas o iterativas

Además se incorporan técnicas:

- Diseño descendente (Top-Down)
- Descomposición de acciones compuestas en otras más simples
- Uso de estructuras de control básicas

9.2. Estructuras de Control de Flujo de Ejecución

- IF
  if (condicion) *sentencia* **else** *sentencia*

  O también:

  if (condicion) *sentencia*
  EQUIVALENTE A
  if (condicion) *sentencia* **else** *NADA*

- SWITCH
  switch (variable) {
  case *valor1: sentencia*
  case *valor2: sentencia*
  ⋮
  default: *sentencia*
  }

  O también:

  switch (variable) {
  case *valor1: sentencia*
  ⋮
  case *valorN: sentencia*
  }

9.3. Estructuras de Repetición

- DO-WHILE

  do *sentencia* **while** (condicion);

- WHILE

  while (condicion) *sentencia*

- FOR

  for (inicializacion**;** condicion**;** incremento**)** *sentencia*

  EQUIVALENTE A

  inicializacion;
  while (condicion) {*sentencia*; incremento;}

  EQUIVALENTE A

  inicializacion;
  if (condicion) **do** {*sentencia*; incremento} **while** (condicion);

9.4. Entrada / Salida por consola

▷ Prototipos en stdio.h

`int getchar(void)`

Lee un carácter de la entrada estándar

`int putchar(int c)`

Escribe un carácter en la salida estándar

`char *gets(char *s)`

Lee una cadena de caracteres de la entrada estándar

`int puts(char *s)`

Escribe una cadena de caracteres en la salida estándar

int printf(char *formato[, argumento1,...])

Imprime la cadena formato sustituyendo cada carácter de conversión (%x) por el argumento que corresponda (por orden)

Caracteres de conversión:

```
%d ó %i: entero decimal
%c: carácter (solo uno)
%s: cadena de caracteres
%e ó %E: numero en punto flotante
         (notación exponencial)
%f: numero en punto flotante (notación decimal)
%u: entero decimal sin signo
%o: entero octal sin signo
%x ó %X: entero hexadecimal sin signo
%p: muestra un puntero
%%: se imprimira %
```

Modificadores y Secuencias de Escape

Las órdenes de formato pueden tener modificadores que especifiquen la amplitud del campo,el número de cifras decimales y un indicador de ajuste a la izquierda

Modificador para entero largo $long\ int$

```
%ld entero largo (32 bits)
```

Secuencias de Escape:

```
\n: nueva línea
\t: tabulador
\b: retroceso
\r: retorno de carro
\f: salto de página
\\: back_slash
\': apóstrofe
\": comillas
```

int scanf(char *formato[,...])

Lee datos de entrada estándar:

```
%d ó %i: entero decimal
%c: carácter (solo uno)
%s: cadena de caracteres
%e: numero en punto flotante
    (notación exponencial)
%f: numero en punto flotante (notación decimal)
%h: short int
%o: entero octal sin signo
%x: entero hexadecimal sin signo
%p: lee un puntero
```

Sujetos a modificadores

Ejemplos:

```
float x = 2.3;
int y = -1;
printf("x = %f\t y = %d\n", x, y);
```