**Programación estruturada**

**Control secuencial**

* As instrucións execútanse na mesma orde na que están escritas.

**Estructuras de control**

* É posible construír calquera programa mediante a combinación dun conxunto reducido de estruturas de control secuenciais, selectivas e repetitivas.
  + **Selectiva:** Úsase para tomar decisións lóxicas. If, if else, switch.
    - Switch case permite intervalos (case a… b)
    - En C, a condición pode ser unha expresión lóxics ou unha variable enteria (0 equiválese co falso)
  + **Repetitivas:** Realizan varias veces o mesmo conxunto de operacións. for, while, do-while
    - O for utilízase cando se coñece o número de repeticións

[ for (iniciación; condición; incremento) ]

* + - Repíteme mentres que a condición sexa certa.
  + Operadores:
    - De comparación: <, >, ==, <=, >=, !=
    - Lóxicos: &&, ||, !
    - Preferencia operadores: ! e funcións, aritmético, de comparación, lóxicos, ?:, de asignación (=, +=, -=, /=, \*=)

**if, else if**

* Execútase se se cumpre a condición especificada.
* Pode ser seguido dunha cantidade indefinida de else if, e pode rematar con un ‘else’ para o caso xeral. Despois do ‘else’ non pode haber outro else if.

**switch(n)**

* Composto por varios apartados **case** valor:, que se executan cando o valor correspóndese co da variable.
* No final posúe un caso **default:** que se executa cando non se cumple ningún dos previos.
* Todos os case deben rematar cun **break**. Do contrario, ao analizar un caso que non ten break, pasa ao seguinte sen considerar a súa condición. O default non precisa break, e cando un case contén só un comando e un break non require {}.

**For**

* Execútase ata que unha variable, que cambia con cada iteración, chega a certo valor.
* Útil para cando o valor de repeticións é coñecido, como ao editar vectores.
* Para matrices, empréganse 2 for aniñados.

**While**

* Execútase ata cumplir unha condición. Se se emprega un do{}while(); executarase unha vez primeiro e despois comprobará a condición.
* Dentro dun bucle while, o predicado que se mantén constante dende inicio a fin chámase **invariante.**
* Por exemplo, nun programa que procura a primeira aparición de x nun vector A[N]: {0 <= i <= N AND x ∉ {A[0],..., A[i-1]}} é o invariante, mentres que {0<= i <= N AND x ∉ {A[0],..., A[i-1]} AND (i== N OR x== A[i]) é a poscondición.
* É o máis axeitado para buscar un elemento nunha lista, pois pode deterse cando o atopa.

**Vectores**

* Declaranse mediante: int v[5], float a1[]={2.0, 4.5, 6.2}
* Operacións: a3[1] = a2[1] + 2\*a1[1]. Non se pode operar entre vectores, só elemento a elemento.
* Só se poden asignar valores a un vector cando se declara inicialmente.
* Na memoria almacénanse en posicións consecutivas na memoria. Un vector de n enteiros ocupa n\*sizeof(int).

**Matrices**

* float mat2[10][24]: define unha matriz 10x24. Os índices comezan en 0
* Os datos almacénanse en memoria na orde de leer a matriz de esquerda a dereita e de arriba abaixo.
  + O espazo no que se almacena é:

(espazo do primeiro elemento + i\*n\_cols + j)\*sizeof(float)

**typedef** tipo\_elemento

* Permite definir un novo tipo de dato. Exemplo: **typedef** tipo\_elemento tipo\_matriz [N][M] (sendo tipo\_elemento o tipo de dato na matriz, por exemplo float ou int)
  + Por exemplo: **typedef** float Matriz10x10[10][10]; Matriz10x10 mat1;
  + define mat1 como unha matriz 10x10.

**Rexistro de datos**

* Represéntase mediante struct nome {tipo\_dato dato1; tipo\_dato dato2;...}
* Emprégase para crear unha estrutura con varios campos de datos.
* Accédese ás variables mediante nome.dato1
* Ao empregalas como parámetro nun procedemento é preciso empregar **struct** e o nome da estructura.
* Un **rexistro xerárquico** é aquel que contén un rexistro como variable.

**Acceso secuencial**

* Acceso á memoria mediante bucles for. Por exemplo, para modificar todos os valores dun vector ou dunha matriz.

**Deseño descendente**

* Estratexia de programación de problemas non triviais.
  + Descompor o problema en subproblemas máis sinxelos
  + Descompor cada subproblema para chegar a accións implementables de forma directa.
  + Exemplo: Considerar variables necesarias, ler o seu valor, comprobar se son válidas, realizar a operación
* Para un esquema repetitivo, identificamos as accións útiles a repetir e as variables necesarias, identificamos como actualizar a información entre iteracións, logo a condición de terminación e finalmente os valores iniciais das variables.
* **Diagrama de fluxo de datos:** Esquema gráfico que expresa o proceso de deseño descendente.
  + Accións expresadas de xeito iterativo e incremental
  + Identifícanse os datos e variables necesarias
  + Especifícanse os entes externos cos que interactúa o programa
  + Formado por un diagrama de contexto (datos de entrada e saída) e varios DFDs en niveis interemedios.
    - Nun esquema, os datos non poden ir entre entidade externa e almacén directamente, debe pasar todo por un proceso
  + **Convencións:**
    - Nomes significativos para procesos, fluxos e almacéns
    - Numerar os procesos cun único número na parte superior.
    - Para precisar máis, pódense expandir en novas páxinas para outros niveis de DFD