**10° lezione prog1**

L'array è una struttura dati omogenea(dati dello stesso tipo) fatta da una sequenza di celle e consecutiva.Inoltre è statico perché la sua dimensione non può cambiare durante l'esecuzione del programma. si parte da zero per contare le posizioni dell’array

Un Array si dichiara nel seguente modo: **int V[10];**

* **Tipo (int float double)**
* **Identificatore**
* **[]**
* **[dimensione]**

Se non si inizializza il vettore non si ha la garanzia che  nelle locazioni predisposte non ci siano valori "sporchi" (sequenze di bit randomiche).

Si utilizzano diversi tipi di inizializzazione:

* int V[10]={1,2,3,4,5,6,7,8,9,10}; In questo modo si ha **una inizializzazione completa  del vettore.**
* int W[10]={1,2,3,4}; In questo modo si ha  una **inizializzazione parziale degli elementi**  e i restanti elementi vengono posti a 0**.**
* int Z[]={1,2,3,4.5,6,7, 8}; In questo modo l**a dimensione viene  definita implicitamente.**

Per inizializzare tutti gli elementi del vettore a 0  ci sono due modi:

1. **int V[10]= {};**
2. **int W[10]={0};**

**La dimensione dell’array può essere definita anche mediante la diretteiva al pre-processore #define. Nel seguente modo: #diefine N int v[N]; o anche nel seguente modo: int N=10; int V[N] (VLA Variable Lenght Array (C99))**

**Le matrici o array a due dimensioni**

Sono formate da dati omogenei (dello stesso tipo) ed hanno due dimensioni righe e colonne. Il numero totale di elementi sarà dato da N X M cioè le righe per le colonne. L’elemento della matrice si indentifica nel seguente modo: M[2][3] così si sta indicando l’elemento di riga 2 e colonna 3.

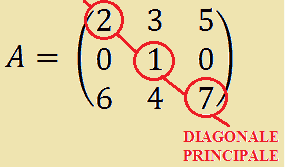
Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

In questo caso si ha una matrice 4 X 11.

Se si devono andare a visitare tutti gli elementi della matrice sono necessari due cicli uno per le righe ed uno per le colonne (FOR). La condizione di questi due cicli deve essere minore N e di M (N-1 ed M-1). I due cicli uno dentro l’altro si chiamano cicli annidati. Si avranno in totale N X M iterazioni.

**Diagonale principale della matrice**



La diagonale principale che si ha solo nelle matrici quadrate. Gli indici della diagonale di sopra sono i seguenti: i=j (0,0 1,1 2,2) quindi si utilizza un solo ciclo for. Perché appunto i due indici sono uguali. Pertanto si avranno N iterazioni.

**Diagonale secondaria della matrice**

**Immagine che contiene Carattere, schermata, numero, design

Descrizione generata automaticamente**



**La diagonale secondaria di una 3 X 3 è alle posizioni: (0,2) (1,1) (2,0) notiamo che la i aumenta e la j diminuisce**. **Sarà pertanto: N-i-1. M[i][N-1-i] ES M[1][3-1-1] M[2][3-1-2].**

**Si dichiarano nel seguente modo: int M[2][2];**

* **Tipo (int float double)**
* **Identificatore**
* **[] per le righe**
* **[] per le colonne**
* **[dimensione righe][dimensione colonne]**

Il compilatore vede la matrice come un insieme di vettori uno dopo l'altro (N X M celle contigue).

Le funzioni operano in un segmento di memoria chiamato **stack**.

Ogni funzione opera in un pezzo di stack che viene allocato prima dell'esecuzione della funzione(1° istruzione della funzione)(Record di attivazione per la funzione main()). tutte le variabili dichiarate sono allocate nel record di attivazione dello stack.



Quando si devono maneggiare grandi quantità di dati conviene utilizzare l’allocazione dinamica. Se non si inizializza la matrice non si ha la garanzia che  nelle locazioni predisposte non ci siano valori "sporchi" (sequenze di bit randomiche).

Si utilizzano diversi tipi di inizializzazione:

* **int Z[4][3]={1,2,3,4,5,6,7,8,9.10,11,12};**

Per inizializzare tutti gli elementi di una matrice a 0 si procede nel seguente mondo:

* **int M[10][2]={0};**
* **int M[10][2]={};**

ATTENZIONE: NON SI Può INIZLIZZARE UNA MATRICE NEL SEGUNTE MODO:

* **int Z[][] = {1,2,3,4,5,6,7,8,9,10,11,12}; perché il compilatore sa che si tratta di un array multidimensionale ma non riesce a capirne la dimensione delle righe.**
* int Z[][M] = {1,2,3,4,5,6,7,8,9,10,11,12}; in questo modo, invece, il compilatore sa che la lunghezza della riga è M quindi va controllare il numero di inizializzatoriAd esempio se M fosse 3 (int Z[][M] = {(1,2,3),(4,5,6,)(7,8,9),(10,11,12)};)

**Numeri pseudocasuali**

Sono applicati in diversi ambiti come per la gestione di magazzini o in ambito scientifico/finanziario. I numeri devono essere randomici, il generatori di numeri casuali deve essere controllabile cioè che con lo stesso input deve essere in grado di generare lo stesso (seme) risultato, deve avere la portabilità (su diverse architetture), ed efficiente intesa come le risorse di calcolo che occupa la propria esecuzione.

Le funzioni rand() ed srand() di stdlib.h

Per utilizzare queste funzioni bisogna utilizzare l’header stdlib.h. e l’header time.h per utilizzare la funzione time().

* **rand()**: restituisce un numero intero e non prende parametri in ingresso. Alla chiamata restituisce il prossimo numero della sequenza. Il numero restituito e’ tra 0 e RAND\_MAX (dell’heder stdlib.h)
* **srand()**:ha come paramento in ingresso(seme o seed). Si utilizza quando si vuole rei inizializzare la sequenza con un certo numero chiamato seed. Non si conosce l’algoritmo che genera numeri casuali ma ci basti sapere che a semi differenti corrispondono sequenze randomiche differenti. Se non si invoca la funzione srand() per default il seme sarà 1. Se si inserisci lo stesso seme si otterrà la stessa sequenza di numeri casuali. Per default. Specificare il seme è come inizializzare la sequenza. La funzione time() restituisce il tempo in secondi percorsi dal 1° gennaio del 1970 e viene utilizzata quando ad esecuzioni della funzione rand() devono restituire sequenze di numeri diverse, in questo modo: srand(time(0)); oppure srand(time(NULL));

**Come generare numeri casuali da 0 a N-1**

Si procede nel seguente modo: **rand()%N (DA 0 A N-1).**

**Come generare numeri casuali in un range [a, b]**

Dati due numeri interi a e b con a<b **rand((b-a+1)+a))**

* Con **b-a+1** si ottiene l’ampiezza dell’intervallo
* Per farlo partire da a e non da 0 si procede nel seguente modo: **((b-a+1)+a))**

Come generare numeri casuali in virgola mobile

Si procede nel seguente modo**: doble n=rand()/(RAND\_MAX\*1.0);**

**Come generare numeri casuali a partire da un range di due numeri in virgola  mobile**

Si prende questo numero compreso tra 0 e 1 si riscala in modo da ottenere un numero compreso tra 0 e Y- X e poi aggiungere X. in codice: **double r=((rand()/RAND\_MAX\*1.0)\*x-y)+x**

* **Numero compreso tra 0 e 1**
* **Numero compreso tra y - x**
* **Numero compreso tra X e Y( aggiungendo X a tutta l'espressione.).**