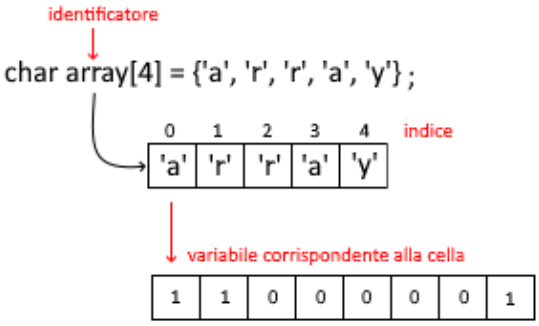
Array

Un array è un insieme di variabili consecutive dello stesso tipo. Come le variabili gli array hanno un identificatore che li rappresenta all'interno del programma ed uno spazio di memoria che ha permette di salvare i valori d'interesse per il programma. Rispetto alle variabili gli array hanno il vantaggio di poter gestire più dati dello stesso tipo sotto il medesimo nome, le componenti dell'array si chiamano celle e ad ognuna corrisponde un indice specifico.



Dichiarazione di un array

Per dichiarare un array all'interno di un nostro programma si utilizza la seguente notazione

```
tipo identificatore[ dimensione fissa ];
```

ad esempio:

```
float f[3];        //f è un array in grado di contenere il valore di 3 numeri reali
char c[11];        //c è un array in grado di contenere il valore di 11 caratteri
int i[];           //i è un array in grado di contenere valori interi ma non ha una grandezza specificata
```

Attenzione: non è possibile utilizzare una variabile al posto di **numeri costanti**, per poter creare dei vettori di dimensione variabile dovremo utilizzare delle strutture chiamate **array dinamici**.

Inizializzazione di un array

Nel paragrafo precedente abbiamo visto come inizializzare degli array di vari tipi, ora ci occuperemo della loro inizializzazione, cioè dell'assegnamento dei loro valori.

Gli array si possono popolare utilizzando le seguenti sintassi:

```
float f[3] = {3.4, 2.5, 1.2};
char c[11] = {'1', '1', 'c', 'a', 'r', 'a', 't', 't', 'e', 'r', 'i'};
bool b[2] = {true, false};
int i[] = {1, 2, 7, 9, 122};
```

Nel caso del vettore in cui dichiariamo un array senza specificarne la dimensione e lo inizializziamo con n valori la sua dimensione sarà uguale ad n, guardando gli esempi precedenti possiamo dire la dimensione di i[] è 5 perché è stato inizializzato con 5 valori.

Inoltre, è bene fare attenzione ai seguenti due casi:

1. Nel caso in cui dichiarassimo un array di grandezza specifica (tipo array[8]) e lo inizializzassimo solo in parte il valore assunto dalle celle restanti sarebbe 0.
   `int array1[8] = {1, 2, 3, 4};` corrisponde a `int array1[8] = {1, 2, 3, 4, 0, 0, 0, 0};`
   Questo implica che se utilizziamo la sintassi
   `int arrayA[6] = {};`
   L'array viene completamente riempito di 0 corrispondendo alla notazione:
   `int arrayA[6] = {0, 0, 0, 0, 0, 0};`
2. Nel caso in cui provassimo ad inizializzare un array con più valori di quelli che abbiamo dichiarato il compilatore ci segnalerà errore.
   `char array2[2] = {'a', 'b', '+', '+'};` *//error: too many initializers*

Utilizzo di un array

Come abbiamo visto gli array possiamo inizializzare facilmente gli array a dei valori precisi, ma come possiamo accedere a quei valori durante l'esecuzione del programma?

Per rispondere a questa domanda dobbiamo ricordare che ad ogni valore corrisponde un indice intero preciso che parte dal numero 0 e si conclude al valore n – 1, inserendo i numeri appartenenti a questo range all'interno delle parentesi quadre possiamo leggere il valore delle celle.

Per esempio, dichiariamo e inizializziamo il seguente array:

```
char a[5] = { 'c', 'i', 'a', 'o', '\0' };
```

Utilizzando la sintassi a[0] possiamo accedere ed utilizzare il valore salvato in quella determinata cella, in questo caso il character literal 'c':

```
cout << a[0];      //Output: c
```

Se aggiungiamo l'utilizzo di un ciclo possiamo **inizializzare** e **visualizzare** tutti i valori appartenenti al vettore fino all'indice *n – 1*, utilizzando un ciclo possiamo scrivere:

```
for( int i = 0; i <= (5 - 1); i++ ){
        cout << "Inserire un valore: ";                    //Inserimento
        cout << v[i];
        cout << "Valore inserito" << v[i] << '\n';         //Visualizzazione
}
```

Esercizi

1. Dichiarare ed inizializzare un array con i primi 10 numeri naturali partendo da 0.
2. Dichiarare ed inizializzare un array con i caratteri che compongono la parola "che bella l'informatica!".
3. Dichiarare ed inizializzare un array con 5 valori booleani a vostra scelta.
4. Dichiarare un array di numeri interi di dimensione 10 e inizializzarlo con numeri inseriti in input da utente.
5. Dichiarare un array di numeri interi di dimensione 5 e inizializzarlo utilizzando un ciclo ed un contatore che parte da 0 ed esegue un incremento di due in due (count = 0, 2, 4, 6, …).
6. Dichiarare un array di caratteri di dimensione 26 e inizializzarlo utilizzando un ciclo ed un contatore che parte dalla lettera minuscola 'a' fino alla lettera 'z'.

   BONUS: inserire in modo alternato una lettera minuscola ed una maiuscola.
7. Dichiarare un array di dimensione 12, far inserire all'utente 12 numeri in input da utente e ordinarli in ordine crescente.
8. Dichiarare un array di dimensione 12 (calcolare la sua dimensione con l'operatore *sizeof*) e inserire al suo interno il valore corrispondente all'indice della cella, inserire contemporaneamente i valori nella prima e nell'ultima cella, poi nella seconda e nella penultima e così via fino ai due indici centrali.
   BONUS: dichiarare un array di dimensione 21 ed eseguire lo stesso esercizio (attenzione alla cella centrale)

Video per la comprensione

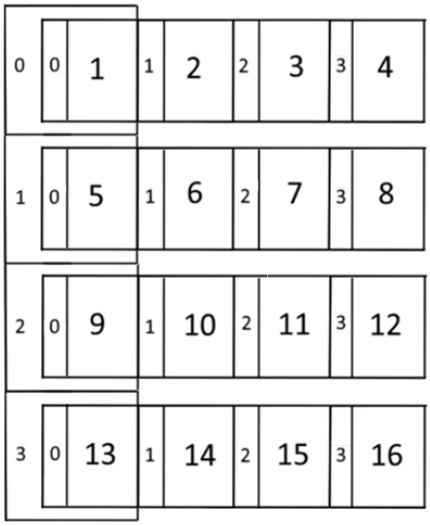
- https://www.youtube.com/watch?v=-TcFY_wX3Uk
- https://www.youtube.com/watch?v=ENDaJi08jCU

Array bidimensionali

https://stackoverflow.com/questions/22950331/implementing-an-n-dimensional-matrix-in-c (dopo)

Nella maggior parte dei linguaggi è possibile dichiarare inizializzare anche degli array a due dimensioni, cioè degli array di array. La sintassi per utilizzare questi costrutti è la seguente:

```
int array[4][4]={ {1,2,3,4}, {5,6,7,8}, {9,10,11,12}, {13,14,15,16} };
```



Dove la **prima cella** indica il **numero degli array** presenti nella matrice mentre la **seconda** indica il **numero di celle per ogni array**.

Altre notazioni:

```
float array2_f[2][3] = { { 1.2,  2.7,  3.1 }, { 4.6,  5.7,  6.8 } };

char array2_c[3][2] = { { 'a',  'b' }, {  'c',  'd' }, { 'e',  'f' } };

bool array2_b[3][3] = { { 1,  0,  1 }, { 1,  1,  0 }, { 0,  0,  1 } };

int array2_a[2][2];                    //array = allocato ma non inizializzato;

int array2_b[2][2] = {};               //array = { {0, 0}, {0, 0} };

int array3_c[2][2] = { { 2, 3} };      //array = { {1, 2}, {0, 0} };
```

Utilizzo di un array bidimensionale

L'utilizzo degli array bidimensionali è il medesimo degli array monodimensionali, l'unica importante differenza è il fatto che per valutare ed elaborare questi tipi di array si devono utilizzare due indici al posto di uno.

Per i prossimi esempi utilizzerò il seguente array

```
int array2[3][2];
```

Esempio 1 – Caricamento di valori successivi in un array bidimensionale

```
int counter = 0;
for(int i = 0; i < 3; i++){
        for(int j = 0; j < 2; j++){
            array2[i][j] = counter;
            counter++;
        }
 }
```

Esempio 2 – Visualizzazione dei valori di un array bidimensionale

```
for(int i = 0; i < 3; i++){
    for(int j = 0; j < 2; j++){
         cout << array2[i][j] << '\t';
    }
    cout << '\n';
}
```

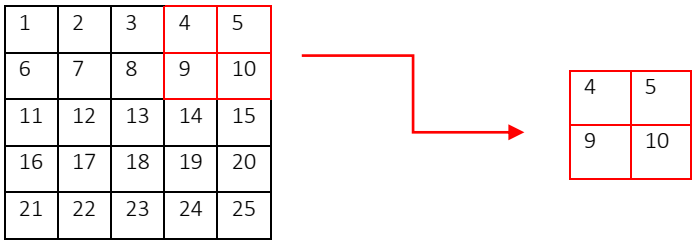
Esempio 3 – Calcolo del numero di colonne e del numero di righe di un array bidimensionale e caricamento di valori successivi

```
int n_colonne = sizeof(array2[0]) / sizeof(int);
int n_righe = sizeof(array2) / (sizeof(int) * n_col);
for(int i = 0; i < n_righe; i++){
        for(int j = 0; j < n_colonne; j++){
            array2[i][j] = counter;
            counter++;
        }
 }
```

Esercizi

Dichiarare ed inizializzare un array bidimensionale di 5 righe e 5 colonne riempendolo successivamente con valori successivi, da 0 a 25.

1. Visualizzare l'array con una formattazione adeguata.
2. Sommare fra loro gli elementi di ogni riga e salvarle i risultati in un altro array, monodimensionale, di 5 celle;
3. Estrarre una matrice 2 x 2 dalla sezione nord-est della matrice, colorata in rosso, e salvare i valori in un array dimensionale della medesima grandezza



4. Calcolare il determinante della matrice trovata sapendo che:

$$det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = (a * d) - (b * c)$$

Per approfondire il determinante di una matrice quadrata vi lascio questi link:

- YouMath
- YouTube

Array n-dimensionali

Ho trovato un progetto GitHub che permette di utilizzare e creare "facilmente" array n-dimensionali, se siete curiosi vi lascio il link qui sotto.

link