(Per una visualizzazione ottimale dei colori attivare il tema scuro)

In **C++** **\*** e **&** vengono utilizzati per gestire i puntatori.

**\*** Viene utilizzato subito dopo un type nella dichiarazione di una variabile per

indicare che la variabile sarà un puntatore di type :

*Esempio di inizializzazione di una variabile:*

double\* numericonvirgola = 0.0;

*Esempio di inizializzazione di un array dinamico:*

int\* array = new int[5]{1,2,3,4,5};

Oppure per dereferenziare una variabile, ovvero, dato un indirizzo di memoria

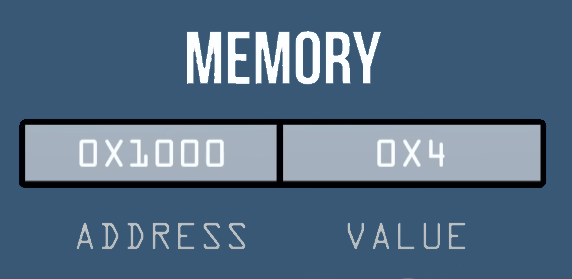
viene restituito il valore a cui quest’ultimo punta.

*(conoscere il type serve a sapere di quanti byte andare avanti in un’istruzione come \*(variabile+1) )*

**&** Viene utilizzato prima del nome di una variabile per restituire l’indirizzo di memoria di quella variabile

*Esempio:*

int a=4;



cout << a; // 4

cout << &a; // 0x1000

*Esempi:*

La variabile intera 10 è salvata nella variabile int numero

all’indirizzo di memoria 0x1008.

Abbiamo una variabile int\* a che contiene 0x1008.

cout << numero; // 10

cout << &numero; // 0x1008

cout << a; // 0x1008 ← indirizzo di memoria di numero

cout << \*a; // 10

cout << &a; // 0x1004 ← indirizzo di memoria di a

(Ricorda: a si trova 4 byte “prima” di numero, nonostante venga dichiarata dopo, perché lo stack funziona appunto come una pila)

cout << \*array; è equivalente a cout << array[0]; // 1

cout << \*(array + 3); è equivalente a cout << array[3]; // 4

cout << \*(array[3]); invece è sempre sbagliato

int & a = array[2]; crea un ALIAS di array[2] chiamato a.

Qualunque modifica ad a viene applicata ad array[2]

e viceversa

delete è richiesto ogni qual volta si presenta la necessità di deallocare una porzione di memoria precedentemente allocata con la keyword new.

*Deallocare la memoria di una variabile:*

delete variabile;

*Deallocare la memoria dinamica di un array:*

delete [] array;

*Gli array dei vari tipi sono tutti pressoché identici, solo gli array di stringhe hanno una piccola particolarità (essendo difatti a loro volta degli array di char) che gli permette di comportarsi come degli array multidimensionali.*

Creiamo un array dinamico di stringhe e vediamo degli esempi.

string\* nc = new string [2]{“Mario”,“Rossi”};

cout << [0][2];

Il primo indice indica l’elemento in posizone 0 elemento dell’array nc // Mario

Il secondo indice indica l’elemento in posizione 2 dell’array di char “Mario” // r