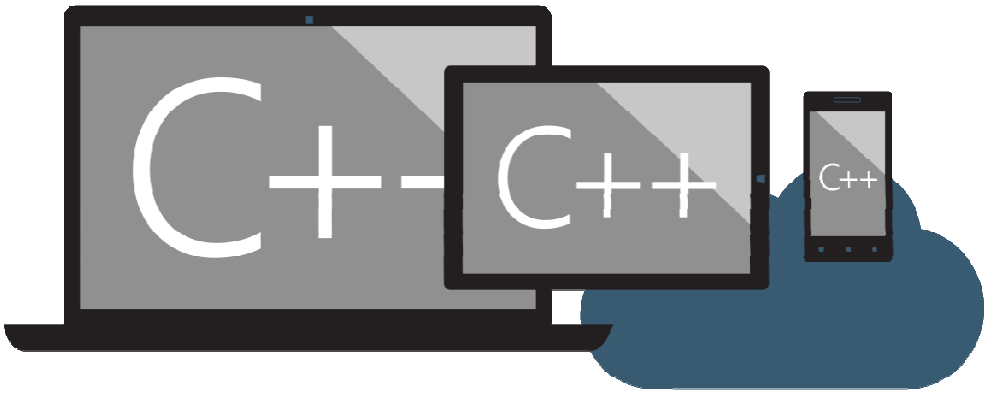
LABORATORIO DI INFORMATICA

Prof.ssa Patrizia Tarantino

1. Modulo 1 – Puntatori in C++
2. Modulo 2 – Introduzione al linguaggio Java

INDICE



MODULO 1 - PUNTATORI

Concetto di PUNTATORE

Una variabile, di qualunque tipo sia, rappresenta un valore posto da qualche parte nella memoria del sistema. Attraverso l'operatore di indirizzamento **(&)**, è possibile ottenere il **puntatore** che rappresenta l'indirizzo di una variabile nellla memoria . Tale valore può essere inserito in una variabile particolare, adatta a contenerlo: una ***variabile puntatore***.

COME SI DICHIARA UN PUNTATORE

La dichiarazione di una variabile puntatore avviene in modo simile a quello delle variabili normali, con l'aggiunta di un asterisco prima del nome.

```
int *p;
```

L'esempio dichiara la variabile ***p*** come **puntatore a un tipo int.**

ASSEGNAMENTO DEL PUNTATORE

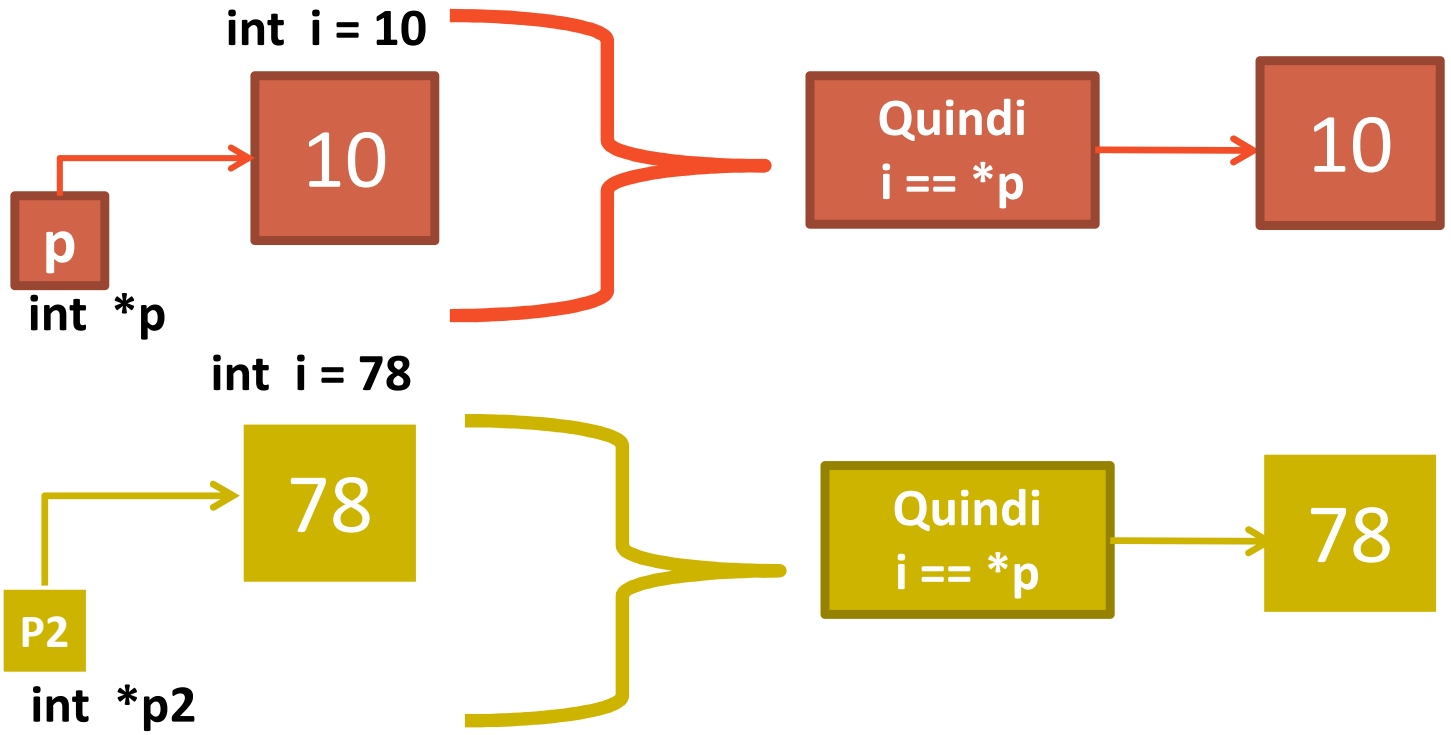
se ***p*** è una variabile puntatore adatta a contenere l'indirizzo di un intero, l'esempio mostra in che modo assegnare a tale variabile il puntatore alla variabile i:.

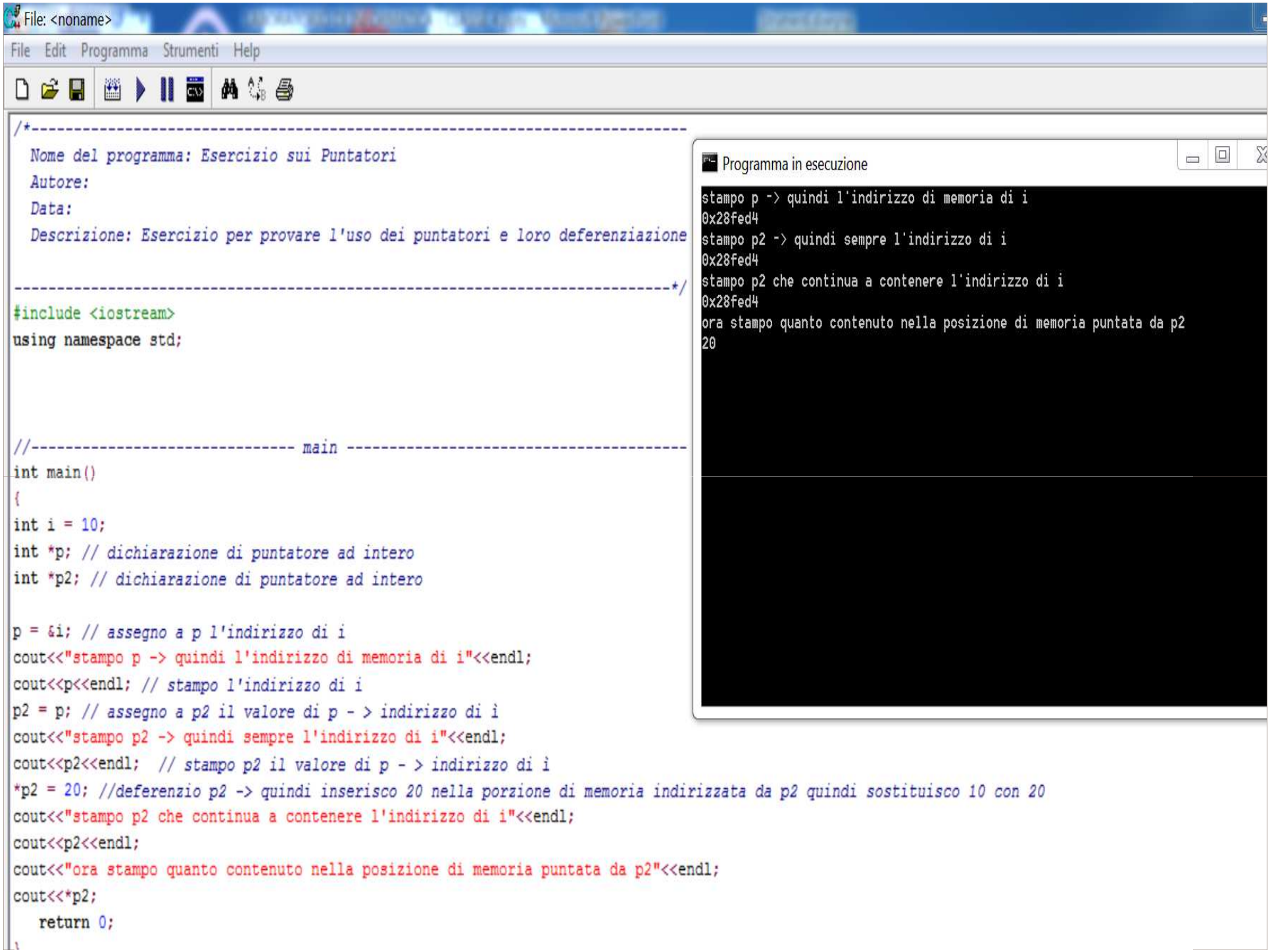
```
int  i = 10;
p = &i;
// L'indirizzo di «i» viene assegnato al puntatore «p».
```

il valore a cui punta la variabile ***p*** è accessibile attraverso l'espressione ****p***

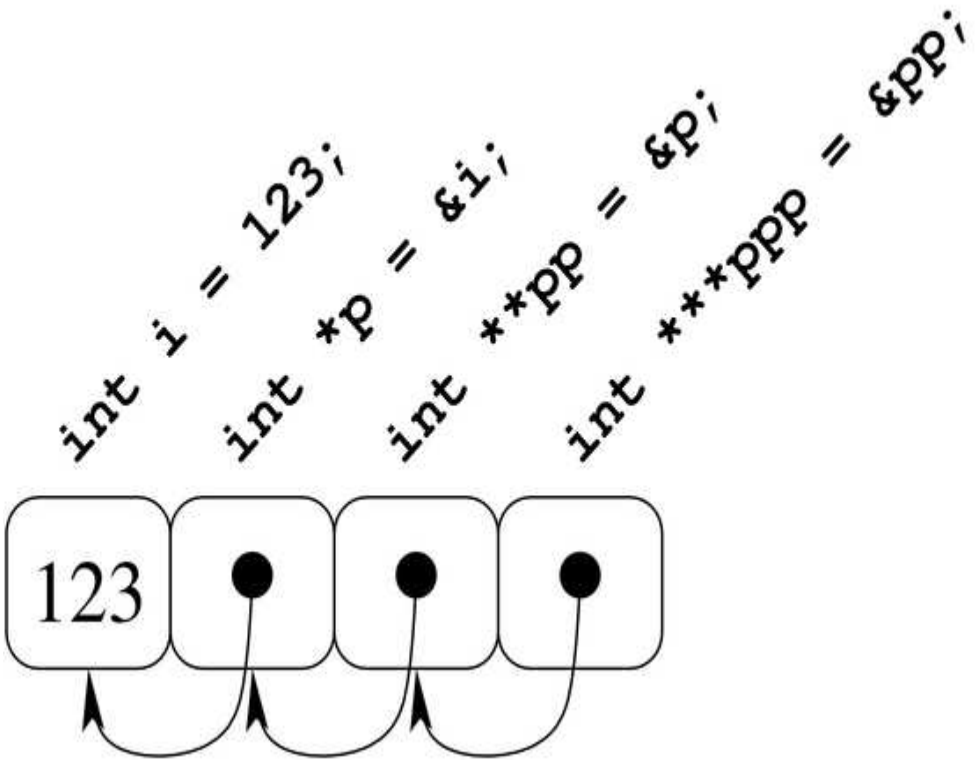
DEFERENZIARE UN PUNTATORE

Attraverso l'operatore di «**dereferenziazione**», **l'asterisco (*), è possibile accedere alla zona di memoria a cui la variabile punta**. Per «dereferenziare» si intende quindi l'azione con cui si toglie il riferimento e si raggiungono i dati a cui un puntatore si riferisce QUINDI:




File: <noname>
File Edit Programma Strumenti Help
/*----------
Nome del programma: Esercizio sui Puntatori
Autore:
Data:
Descrizione: Esercizio per provare l'uso dei puntatori e loro deferenziazione
----------*/
#include <iostream>
using namespace std;
//---------- main ----------
int main()
{
int i = 10;
int *p; // dichiarazione di puntatore ad intero
int *p2; // dichiarazione di puntatore ad intero
p = &i; // assegno a p l'indirizzo di i
cout<<"stampo p -> quindi l'indirizzo di memoria di i"<<endl;
cout<<p<<endl; // stampo l'indirizzo di i
p2 = p; // assegno a p2 il valore di p - > indirizzo di ì
cout<<"stampo p2 -> quindi sempre l'indirizzo di i"<<endl;
cout<<p2<<endl; // stampo p2 il valore di p - > indirizzo di ì
*p2 = 20; //deferenzio p2 -> quindi inserisco 20 nella porzione di memoria indirizzata da p2 quindi sostituisco 10 con 20
cout<<"stampo p2 che continua a contenere l'indirizzo di i"<<endl;
cout<<p2<<endl;
cout<<"ora stampo quanto contenuto nella posizione di memoria puntata da p2"<<endl;
cout<<*p2;
return 0;
}
Programma in esecuzione
stampo p -> quindi l'indirizzo di memoria di i
0x28fed4
stampo p2 -> quindi sempre l'indirizzo di i
0x28fed4
stampo p2 che continua a contenere l'indirizzo di i
0x28fed4
ora stampo quanto contenuto nella posizione di memoria puntata da p2
20

PUNTATORI CHE PUNTANO A PUNTATORI




```
i == *p

i == **pp

i == ***ppp
```


MODULO 2 - INTRODUZIONE AL LINGUAGGIO JAVA

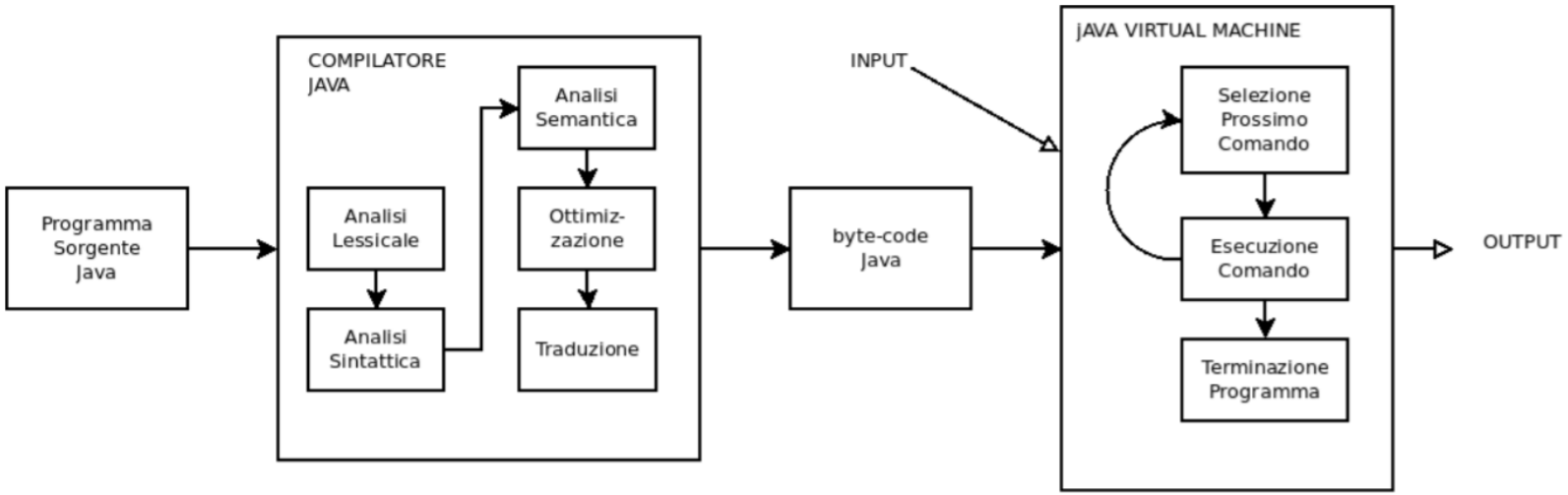
Introduzione al linguaggio Java

Il linguaggio **Java** si basa su un approccio che combina la compilazione e l'interpretazione .

Il programma .java, viene compilato ottenendo un il **byte-code,** questo formato intermedio è indipendente dall'architettura e può quindi essere trasportato su piattaforme diverse. Questo codice, che non può essere eseguito da una macchina reale**, viene a sua volta interpretato da**

Java Virtual Machine (JVM) che lo esegue istruzione per istruzione.

L'approccio compilazione+interpretazione schematicamente:



AMBIENTE DI PROGRAMMAZIONE

Tra i numerosi ambienti di sviluppo, il piu noto è JDK di ORACLE. Questo ambiente è privo di Interfaccia grafica e tutte le operazioni di compilazione ed interpretazione sono inviati da riga di comando.

L'insieme delle librerie fornite dall'ambiente di programmazione si dicono **API**

LE LIBRERIE E LE API

L'ambiente di programmazione contiene varie librerie con metodi e classi che facilitano lo sviluppo di applicazioni riusabili.
Le principali librerie:

- java-lang -> collezione base di classi sempre presente
- java-io-> per gestire i file e i flussi di I/O
- Java-awt-> per gestire i componenti grafici
- Java-net-> per creare applicazioni per il web
- Java.util-> per gestire array dinamici, date, ecc

In Java, la **GESTIONE DELLA MEMORIA** è effettuata automaticamente dal sistema di run-time, che si occupa quindi sia di allocare la memoria che di deallocarla, a differenza di quanto accade con il linguaggio C++ dove è il programmatore a doversi occupare della gestione della memoria attraverso l'uso dei puntatori, con il conseguente rischio di incorrere in errori.