Programmazione Procedurale

Elementi di Base
Dichiarazione dei Dati

versione 3.1

Sommario

- Panoramica
- Identificatori
- Tipizzazione
- Tipi di Dato di Base del C++
- Costanti
- Variabili
- Errori nella rappresentazione

Panoramica

- Nel seguito
  - ⇨ci occupiamo della dichiarazione dei dati in un programma
- Concetti introdotti
  - ⇨identificatore
  - ⇨tipo di dato e tipizzazione
  - ⇨costante e variabile
- Il resto nelle lezioni successive

Panoramica

- Dati principali di un programma
  - ⇨costanti
  - ⇨variabili
- Sono contraddistinti da
  - ⇨un nome (“identificatore”)
  - ⇨un tipo, che ne definisce le caratteristiche – ATTENZIONE: concetto importante

Identificatori

ATTENZIONE
alla sintassi degli identificatori

- Nome di un oggetto del programma
  - ⇨esempi: `pigreco, raggio, cerchio`
- Regole sintattiche
  - ⇨sequenza composta da lettere e/o cifre e/o dal carattere "underscore" '_':
  - ⇨primo carattere: lettera oppure underscore
  - ⇨ATTENZIONE: codice ASCII a 7 bit

  esempi: `mio_prog1, _1bis, Cerchio`

  ~~`città, 1prova, codice libro, m.n`~~

Identificatori

- Sensibili alle maiuscole
  - ⇨`cerchio, Cerchio, CERCHIO`

    sono considerati nomi diversi
- Lunghezza massima
  - ⇨dipende dal compilatore
  - ⇨è opportuno trovare un compromesso tra leggibilità e lunghezza Es: `misuraDellaSuperficieDelCerchio`
- Identificatori riservati
  - ⇨parole chiave del linguaggio

    Es: `main, const, float, cin`

Tipizzazione

ATTENZIONE
concetto
importante

- Tipizzazione
  - ⇨strategia con la quale un linguaggio di programmazione definisce le caratteristiche di un dato
- Ovvero
  - ⇨come rappresentare il valore in memoria
  - ⇨l'insieme dei valori che può assumere
  - ⇨quali operazioni consentire

Tipizzazione

- Insieme dei valori consentiti
  - ⇨int raggio; → numeri interi
  - ⇨string nome; → sequenze di caratteri
- Spazio di memoria associato al dato
  - ⇨int raggio; → 32 bit
  - ⇨char c; → 8 bit
- Insieme di operazioni consentite sul dato
  - ⇨somma sui numeri, concatenazione di stringhe

Tipizzazione

- Le principali strategie di tipizzazione
  - ⇨tipizzazione statica o dinamica
  - ⇨tipizzazione forte o debole
- Tipizzazione statica
  - ⇨ogni dato ha un tipo chiaramente dichiarato nel codice, che non può cambiare
- Tipizzazione dinamica
  - ⇨i dati vengono utilizzati senza associargli un tipo statico, hanno un tipo, ma questo può cambiare

Tipizzazione

- Tipizzazione forte
  - ⇨il linguaggio adotta regole rigide sull'utilizzo dei dati di tipi diversi
  - ⇨es: non si può sottrarre una stringa da un numero
- Tipizzazione debole
  - ⇨il linguaggio non ha regole rigide, e cerca di "cambiare" i tipi per effettuare le operazioni

Tipizzazione

- Il C++/Clean C
  - ⇨è un linguaggio staticamente tipato
  - ⇨con regole di tipizzazione mediamente forti
  - ⇨analogamente Java (più fortemente tipato)
- NOTA
  - ⇨non tutti i linguaggi sono dello stesso tipo
  - ⇨es: JavaScript e Python

Tipizzazione

- Di conseguenza
  - ⇨in C/C++ un dato deve essere "dichiarato" prima di poter essere usato
  - ⇨nella dichiarazione deve essere specificato il suo "tipo di dato"

```
// C++
int raggio;
float cerchio;
raggio = 1;
```

```
// JavaScript
let raggio;
let cerchio;
raggio = 1;
```

```
// Python
raggio = 1
cerchio = ...;
```

Tipi di Dato di Base

ATTENZIONE
-2^{n-1} e $2^{n-1}-1$

- Tipo intero: int
  - ⇨esempio: `int raggio;`
  - ⇨32 bit di memoria in complemento a 2
  - ⇨valori (-2.147.483.648, +2.147.483.647)
- Esempi di valori ammissibili
  - ⇨15
  - ⇨256
  - ⇨-46
  - ⇨765432

Tipi di Dato di Base

- Tipo reale: float
  - ⇨esempio: `float cerchio;`
  - ⇨32 bit di memoria in virgola mobile
  - ⇨valori in modulo tra (1.2×10^{-38},3.4×10^{38})
- Esempi di valori ammissibili
  - ⇨12.67
  - ⇨-13.0
  - ⇨0.234
  - ⇨.234
  - ⇨.345E+4 (notazione esp.: $0.345 \times 10^4 = 3450$)

Tipi di Dato di Base

- Tipo reale doppia precisione: double
  - ⇨esempio: `double portataAla;`
  - ⇨64 bit di memoria in virgola mobile
  - ⇨valori in modulo tra (1.7×10^{-308},1.7×10^{308})
- Esempi di valori ammissibili
  - ⇨12.67
  - ⇨-13.0
  - ⇨.234
  - ⇨.345E-4 (notazione esponenziale)

Tipi di Dato di Base

- Tipo carattere: char
  - ⇨esempio: `char segnoSchedina;`
  - ⇨8 bit di memoria in codice ASCII
  - ⇨lettere, cifre e simboli
- Esempi di valori ammissibili
  - ⇨‘a’, ‘X’, ‘0’, ‘8’, ‘?’, ‘+’, ‘(‘, ‘)’ ...
  - ⇨‘ ’, ‘\n’, ‘\t’ (caratteri speciali: “spazi bianchi”)
- Attenzione: codice ASCII a 7 bit
  - ⇨non sono incluse i simboli estesi (accentate)

Tipi di Dato di Base

- Attenzione
  - ⇨char vale per i singoli caratteri, non per le "stringhe di caratteri"
  - ⇨a questo scopo esiste in C++ il tipo string
  - ⇨esempio: `string nome;`
  - ⇨consente di rappresentare sequenze di car.
- Esempi
  - ⇨"Pinco", "Pinco Pallino", "C3PO-!?!", "x", ""
- Non è un tipo di dato semplice (>>)

Tipi di Dato di Base

- Tipo booleano: bool
  - ⇨esempio: `bool trovato;`
  - ⇨1 bit di memoria
  - ⇨due soli valori (vero, falso)
- Esempi di valori ammissibili
  - ⇨true oppure 1
  - ⇨false oppure 0
- Viene utilizzato per controllare se opportune condizioni sono verificate (>>)

Costanti

- Costante
  - ⇨dato che assume lo stesso valore per tutta la durata della esecuzione del programma
- Ne esistono due tipi
  - ⇨valori costanti: numeri, caratteri, stringhe utilizzati nel programma
  - ⇨costanti simboliche: valori costanti a cui viene attribuito un nome

Dichiarazione di Costanti

- Costanti ordinarie (valori)
  - ⇨numeri interi: 15 -20
  - ⇨numeri non interi: -12.345 .345E+2
  - ⇨caratteri: ‘a’, ‘A’, ‘*’, ‘0’
  - ⇨stringhe: “Inserisci il raggio”, “Valore: ”, “a”
  - ⇨booleani: true, false
- Costanti simboliche
  - ⇨dichiarate attraverso la parola chiave “`const`”

Dichiarazione di Costanti Simboliche

- Sintassi

  `const <tipo> <nome> = <valore>;`
- Dove
  - <tipo> è uno dei tipi di dato del linguaggio
  - <nome> è un identificatore
  - <valore> è il valore della costante; deve essere compatibile con il tipo di dati
- Esempi
  - `const int segniTotocalcio = 13;`
  - `const char lettera = 'A';`

Dichiarazione di Costanti Simboliche

- Semantica
  - allocazione di uno spazio di memoria compatibile con il tipo di dato (es: `float` 32 bit)
  - il valore viene memorizzato nello spazio di memoria (es: `3.14`) e non può essere cambiato durante l'esecuzione del programma
  - è possibile riferirsi al valore attraverso il nome

    es: `circonf = 2 * pigreco * raggio;`

    equivale a: `circonf = 2 * 3.14 * raggio;`
- Vantaggio
  - parametricità del programma rispetto al valore

Variabili

- Variabile
  - ⇨dato che può assumere valori differenti durante l'esecuzione del programma
  - ⇨uno spazio nella memoria con un nome
  - ⇨deve essere dichiarata ("annunciata") prima di essere utilizzata nel programma
- Esempi:
  - ⇨`int raggio;`
  - ⇨`float cerchio, circonf;`

Dichiarazione di Variabili

- Sintassi

  `<tipo> <nome>;`

  `<tipo> <nome1>, <nome2>, ...;`
- Dove
  - ⇨<tipo> è uno dei tipi di dato del linguaggio
  - ⇨i nomi sono identificatori distinti (e devono obbedire alle regole sintattiche relative)

Dichiarazione di Variabili

- Semantica
  - ⇨allocazione di uno spazio di memoria della dimensione prevista per il tipo di dato
  - ⇨allo spazio di memoria è associato il nome (identificatore)
  - ⇨lo spazio di memoria può, durante l'esecuzione del programma, contenere valori differenti

Dichiarazione di Variabili: Esempi

- Esempi

```
int contatore;
```

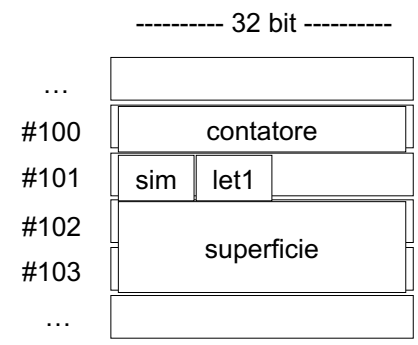
un registro da 32 bit

```
char sim, let1;
```

porzione di registro (8 bit)

```
double superficie;
```

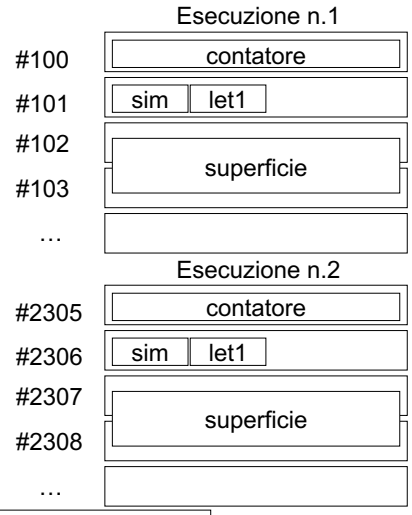
due registri (64 bit)



Dichiarazione di Variabili: Esempi

- Attenzione
  - ⇨l'assegnazione dello spazio vale per una sola esecuzione del programma
- Esempi

```
int contatore;
char sim, let1;
double superficie;
```



Variabili

- Per lavorare con una variabile
  - ⇨il processore ha bisogno di conoscerne lo spazio relativo nella memoria (da quale a quale bit)
  - ⇨ovvero l'indirizzo del primo bit di spazio occupato dalla variabile ed il relativo tipo
  - ⇨Es: contatore: bit 0 del registro #100, 32 bit successivi
  - ⇨Es: let1: bit 8 del registro #101, 8 bit successivi

Variabili

- Per il programmatore invece
  - ⇨è più semplice fare riferimento alla variabile attraverso il suo nome
- Utilizzo della variabile
  - ⇨il processore traduce i riferimenti alla variabile fatti nelle istruzioni attraverso il suo nome nel corrispondente indirizzo
  - ⇨attraverso una tabella nome-indirizzo mantenuta durante l'esecuzione del programma

Inizializzazione delle Variabili

ATTENZIONE al problema dell'inizializzazione

- Attenzione
  - ⇨il valore di una variabile è inizialmente "indefinito"
  - ⇨dipende dal contenuto (casuale) del registro di memoria assegnato alla variabile (0 o 1)
  - ⇨per usare una variabile è necessario attribuirle un valore iniziale
  - ⇨operazione di inizializzazione (= prima assegnazione)

Inizializzazione delle Variabili

```
#include <iostream>
using namespace std;
int main () {
  int num;
  cout << num << endl;
  return 0;
}
```

Schermo

1a esecuzione: **14734480**

2a esecuzione: **-332388**

il valore restituito è casuale

Inizializzazione delle Variabili

```
#include <iostream>
using namespace std;
int main () {
  int num;
  num = 5;
  cout << num << endl;
  return 0;
}
```

Schermo

5

Il valore restituito è predicibile

Inizializzazione delle Variabili

- Sintassi alternativa della dichiarazione
  - ⇨è possibile effettuare l'inizializzazione assieme alla dichiarazione (due istruzioni in una)
- Sintassi

```
<tipo>  <nome> = <valore>;
```

- Esempio

```
int main () {
 int num = 5;
 cout << num << endl;
 return 0;
}
```

ATTENZIONE: differenza con le costanti: si tratta solo di un valore iniziale, che poi può cambiare

Errori nella Rappresentazione

- Dati del programma
  - ⇨costanti e variabili
  - ⇨vengono utilizzate per rappresentare i dati del problema da risolvere
- Attenzione
  - ⇨a causa dei vincoli dell'implementazione ci possono essere errori nella rappresentazione
  - ⇨trabocco
  - ⇨errori dovuti ad approssimazione

Trabocco

- Ogni tipo ha un insieme limitato di valori
  - ⇨è scorretto assegnare alle variabili valori al di fuori dell'intervallo corrispondente al tipo
  - ⇨in questo caso si verifica un fenomeno di trabocco ("overflow")
- In particolare
  - ⇨l'errore non viene segnalato dal processore
  - ⇨il valore nel registro di memoria risulta essere completamente diverso da quella atteso

Trabocco

- Esempio

```
int i; float f;
i = 4000000000;
f = .12e+40;
cout << "i = " << i << endl;
cout << "f = " << f << endl;
cout << "Fine";
```

valore risultante dal tentativo di rappresentare 4 miliardi (numero di 33 bit) su 32 bit (il bit più significativo viene perduto)

Schermo

i = -294967296
f = +INF
Fine

valore speciale utilizzato per segnalare l'overflow in un registro in virgola mobile

Trabocco

- Di conseguenza
  - ogni volta che in un linguaggio è necessario utilizzare un dato, bisogna selezionare un tipo
  - il tipo deve essere sufficientemente generale per contenere i valori del dato
  - e deve essere sufficientemente economico da non rendere inefficienti le operazioni

Errori di Approssimazione

- Rappresentazione in virgola mobile
  - inerentemente approssimata per i numeri reali
- Esempio

```
double x;
double y;
x = 1 / 3.0;   // x vale 1/3
y = 3 * x - 1; // y dovrebbe valere 0
cout << y << endl;
```

Schermo

-5.55112e-17

Errori di Approssimazione

- In effetti
  - ⇨l'intervallo di rappresentazione di float e double contiene dei "vuoti"
- Esempio: float
  - ⇨valori in modulo tra (10^{-38}, 10^{38})

inoltre, NON tutti i numeri reali in questi intervalli sono rappres.

Riassumendo

- Dati di un programma
  - ⇨variabili
  - ⇨costanti
- Tipizzazione e Tipi di dato
- Elementi sintattici introdotti
  - ⇨identificatori (ATTENZIONE)
  - ⇨istruzioni di dichiarazione
- Semantica

Termini della Licenza