Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

Fondamenti di Informatica

Laurea in
Ingegneria Civile e Ingegneria per l'ambiente e il territorio

Linguaggio C: Introduzione

Stefano Cagnoni e Monica Mordonini

Struttura di un progamma C

- La struttura di un programma C è definita

<programma>::=
{<unità-di-codifica}
<main>
{<unità-di-codifica}

Struttura di un programma C

- La parte <main> è obbligatoria ed è definita

<main>::=
main(){
[<dichiarazione-e-definizioni>]
[<sequenza-istruzioni>]
}

Struttura di un programma C

- <dichiarazioni-e-definizioni>
  - introducono i nomi di costanti, variabili, tipi definiti dall'utente
- <sequenza-istruzioni>
  - sequenza di frasi del linguaggio ognuna delle quali è un'istruzione
- Il *main()* è una particolare unità di codifica *(una funzione)*

Caratteri e identificatori

- Set di caratteri
  - caratteri ASCII
- Identificatori
  - sequenze di caratteri tali che

<Identificatore>::=
<Lettera>{<Lettera>|<Cifra>}

Commenti

- Sequenze di caratteri racchiuse fra /* e */
  - /*pippo*/
- Non possono essere innestati

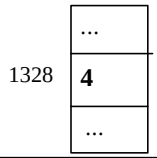
Variabile

- E' un'astrazione della cella di memoria
- Formalmente, è un simbolo associato ad un indirizzo fisico

Simbolo	indirizzo
X	1328

Variabile

- L'indirizzo fisico è fisso e immutabile, cambia il suo contenuto cioè il valore di x
- esempio: x=4



Definizione di variabile

- E' la frase che introduce una nuova variabile
- identificata da un dato simbolo
- e atta a denotare valori di un ben preciso tipo

Esempi

- Definizione di una variabile
- <tipo><identificatore>
- int x; /* deve denotare un valore intero*/
- float y; /* deve denotare un valore reale*/
- char ch; /* deve denotare un valore carattere*/

Tipo di dato

- Esprime in modo sintetico
  - un insieme di valori
  - la loro rappresentazione in memoria
  - un insieme di operazioni ammissibili
- Permette di effettuare controlli statici (al momento della compilazione) sulla correttezza del programma

Esempio

- `int`
  - viene memorizzato con un numero fisso e prestabilito di bit (ad esempio 32 bit)
  - operazioni consentite: *+, -, *, /, %*
  - esistono i modificatori**:**
    - *short*
    - *long*
    - *signed*
    - *unsigned*

Esempi

- `char`
  - può assumere un qualunque codice ASCII (es. 'A' 'c' '2' '!'), tra cui i codici di controllo; i più importanti sono:
    - *'\n'* *a capo*
    - *'\t'* *tabulazione orizzontale*
  - occupa un byte
  - sono consentite le normali operazioni aritmetiche
  - esistono i modificatori:
    - *signed*
    - *unsigned*

Esempi

- `float e double`
  - assumono valori reali, con segno (es. 3.14 -1.7e-6)
  - esiste il tipo modificato *`long double`*

Numeri reali nella macchina virtuale C

- *Float (IEEE-32; 4byte) / Double (IEEE-64; 8 byte)*
  - 1 bit (0=positivo, 1=negativo) per il segno del numero
  - 8 bit/11 bit per l'esponente, dove:
    - i valori da 127 a 254 / da 1023 a 2046 rappresentano gli esponenti positivi da 1 a 128 / da 1 a 1024
    - i valori da 1 a 125 / da 1 a 1021 rappresentano gli esponenti negativi da -125 a -1 / da -1021 a –1
    - i valori estremi 0 e 255 / 0 e 2047 sono riservati
  - 23 bit / 52 bit per la codifica della mantissa

Inizializzazione di variabili

- E' possibile specificare il valore iniziale di una variabile
- <tipo><identificare> = <espr> ;
- int x = 32;
- double speed = 124,6;
- double time = 71,6;
- double km = speed*time; /* inizializzare con una espressione*/

Caratteristiche delle variabili

- Campo d'azione (scope): è la parte di programma in cui la variabile è nota e può essere usata
- Tipo: specifica la classe di valori che la variabile può assumere (e quindi gli operatori applicabili)
- Tempo di vita: l'intervallo di tempo in cui rimane valida l'associazione simbolo/cella di memoria
- Valore: è rappresentato (secondo la codifica adottata) nell'area di memoria associata alla variabile

Valutazione di Espressioni

- Il C è un linguaggio basato su espressioni
- Una espressione è una notazione che denota un valore mediante un processo di valutazione
- Una espressione può essere semplice o composta
- Vi sono operatori relazionali aritmentici e logici

Valutazione in corto circuito

- La valutazione dell'espressione cessa appena si è in grado di determinare il risultato

Valutazione in corto circuito- esempi

- 22||x
  - già vera perchè 22 vero
- 0 &&x
  - già falsa perché 0 è falso
- a||b||c
  - se a||b è vero il secondo non viene neanche valutato
- a &&b && c
  - se a &&b è falso, il secondo && non viene neanche valutato

Operatori infissi, postfissi e prefissi

- Dove posizionare l'operatore?
  - Prima (notazione prefissa)
  - Esempio: + 3 4
  - Dopo (notazione postfissa)
  - Esempio: 3 4 +
  - In mezzo (notazione infissa)
  - Esempio 3 + 4

Priorità degli operatori - Esempi

- Notazione postfissa: 4 5 6 + *
  - si legge come 4*(5+6)=44
- Notazione prefissa: * + 4 5 6
  - si legge come (4+5)*6=54
- nella notazione infissa la priorità degli operatori e la proprietà associativa (che determinano l'uso delle parentesi) è quella a cui siamo abituati

Problema e metodologie di progetto

Il problema del progetto di una soluzione

- Dato un problema non si deve iniziare subito a scrivere il programma per
  1. evitare errori introvabili
  2. evitare di scrivere codice per trovare poi una soluzione migliore e più efficiente
  3. per poter in seguito modificare il programma

Problema e Algoritmo

- Come trovare l'algoritmo "giusto"?
- Occorre distinguere fra due dimensioni progettuali:
  - programmazione in piccolo
  - programmazione in grande
- E seguire questi due principi cardine:
  - procedere per livelli di astrazione
  - garantire al programma strutturazione e modularità

Metodologie di progetto

- Top-down
  - procede per decomposizione del problema in sotto-problemi, per passi di raffinamento successivi, fino a raggiungere problemi risolubili in mosse elementari
- Bottom-up
  - procede per composizione di componenti e funzionalità elementari fino alla sintesi dell'intero algoritmo

Programmazione Top-Down

- La programmazione top-down è una tecnica di programmazione per la realizzazione di programmi con la scomposizione iterativa di un problema in sotto-problemi.

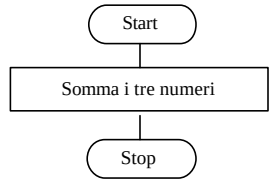
Tecniche di programmazione

- Programmazione *top-down*:
  - scomposizione iterativa del problema in sottoproblemi
  - i sottoproblemi devono essere indipendenti ed avere interfacce ben definite
  - visibilità dei dettagli di ogni sottoproblema solo all'interno del sottoproblema stesso

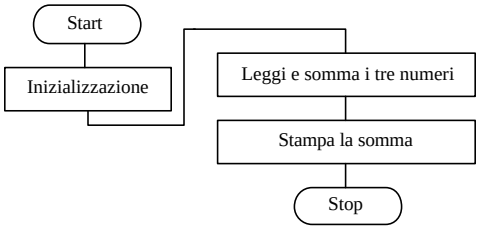
Programmazione Top-Down

- La programmazione top-down si presta molto bene per la definizione di algoritmi con i diagrammi di flusso:
  - All'inizio il diagramma di flusso è rappresentato da un nodo che rappresenta la soluzione al problema.
  - Questo nodo viene scomposto in una rete di nodi in modo iterativo.
  - La scomposizione termina quando i singoli nodi possono essere rappresentati da semplici sequenze di istruzioni del linguaggio di programmazione scelto.

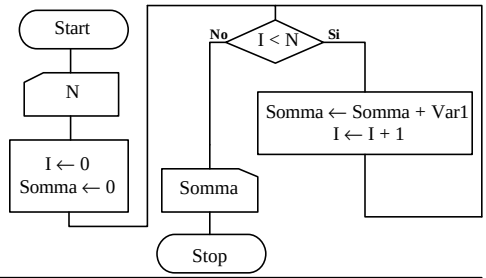
Esempio: Somma di Tre Numeri



Esempio: Somma di Tre Numeri



Esempio: Somma di N Numeri



Esempio

- Problema
  - data una temperatura espressa in gradi Celsius, calcolare il corrispondente valore espresso in Fahrenheit
- Approccio:
  - si parte dal problema e dalle proprietà nel dominio di dati
- Specifica della soluzione:
  - relazione tra grandezze esistenti nello specifico dominio applicativo
  - c*9/5=f-32

Esempio-l'algoritmo

- Dato c
- calcolare f sfruttando la relazione f=32+c*9/5
- solo a questo punto si effettua la codifica

Un possibile programma in C

```
main(){
float c=18; /*Celsium*/
float f=32+c*9/5;
}
```

Nb l'impaginazione serve a noi per leggere il programma, in C le istruzioni sono separate da ";"

Esempio

- Problema
  - dati 3 valori interi>0 (a, b, c) dire se essi possono rappresentare i lati di un triangolo e, in tal caso, di che triangolo si tratta

Esempio

- Si ha un triangolo se e solo se vale a+b>c
- nel caso ciò sia vero, il triangolo è
  - equilatero se a=b e b=c
  - isoscele se a=b (o b=c) ma a≠c
  - scaleno se a≠b≠c≠a

Esempio

- triangolo se e solo se vale a+b>c
  - equilatero se a=b e b=c
  - isoscele se a=b (o b=c) ma a≠c
  - scaleno se a≠b≠c≠a

Espressione C che riassume il tutto:

```
(a+b)<=c ?'n' :
(a==b)&&(b==c) ? 'e' :
(a==b) || (b==c) ? 'i' : 's'
```

Nb non può essere a=c≠b perchè a+b<=c

Un possibile programma in C

```
main(){
int a=3, b=8, c=20;
char ris =
        (a+b) <=c ? 'n' :
                (a==b) && (b==c) ? 'e':
                        (a==b) || (b==c) ? 'i' : 's';
}
```

nb nota l'impaginazione

Costruzione di una applicazione

- Si deve compilare il file (o i files) che contengono il testo del programma (*file sorgente, estensione .c)*
- Il risultato sono uno o più file *oggetto (estensione .o o .obj)*
- si deve collegare i file oggetto l'uno con l'altro e con le librerie di sistema al fine di creare un unico file *eseguibile (estensione .exe o nessuna o a.out)*

Perchè?

- L'elaboratore capisce solo un linguaggio macchina
- il nostro programma opera su una macchina rivestita del sistema operativo che controlla le periferiche (stampante,..)
- alcune istruzioni complesse potrebbero essere dei mini-programmi forniti insieme al compilatore che le ingloba quando occorre

Librerie di sistema

- Insieme di componenti software che consentono di interfacciarsi col sistema operativo usare le risorse da questo gestite e realizzare alcune "istruzioni complesse" del linguaggio

Eseguire un programma

- Una volta scritto e compilato e collegato (linker) lo si può lanciare sull'elaboratore
- e se non funziona?
- Debugger: strumento in grado di eseguire passo passo il programma, vedendo le variabili e al loro evoluzione e seguendo le funzioni via via chiamate

Ambienti integrati di programmazione

- Automatizzano la procedura di compilazione e link dei file
- Possono lanciare il programma sulla macchina e visualizzare l'output a video
- Hanno incorporato le funzioni di debug

Noi utilizzeremo il **turboC3.2 della Borland** presente nei laboratori di base

Debugger

- E' possibile
  - eseguire il programma riga per riga entrando anche dentro le funzioni chiamate
  - oppure eseguire fino alla riga desiderata
  - controllare istante per istante quanto vale una variabile
  - vedere istante per istante le funzioni attive