Fondamenti di Informatica e Laboratorio T-AB
Ingegneria Elettronica e Telecomunicazioni

Lab 03

Scanf e tempo di vita delle variabili

Obiettivo dell'esercitazione

- Chiarire il concetto di tempo di vita delle variabili
- Acquistare familiarità con l'istruzione di input "scanf"

Campo di visibilità

Una variabile è visibile (accessibile):

- All'interno del blocco in cui è definita
- All'interno di ogni sottoblocco

Cos'è un blocco?

- Una istruzione complessa…
- …che raggruppa altre istruzioni e causa la loro esecuzione in sequenza

Esercizio 1

```
int main() {
	int x = 0;
	{
		int y = 1;
		int z = x + y;
	}
	int w = x + y;
}
```

Compila? Perché?

Esercizio 1 - Soluzione

Non compila! Perché “y” è definita dentro un blocco ed è visibile solo al suo interno.

```
int main() {
	int x = 0;
	{
		int y = 1;
		int z = x + y;
	}
	int w = x + y;
}
```

Esercizio 2

```
int main() {
	int x = 0;
	{
		int x = 1;
		{
			int y = x + 1;
		}
	}
}
```

Quanto vale “y” alla fine? Perché

Esercizio 2 - Soluzione

Vale 2, perchè la seconda definizione di “x” copre quella precedente all’interno del blocco.

```
int main() {
	int x = 0;
	{
		int x = 1;
		{
			int y = x + 1;
		}
	}
}
```

Esercizio 3

```
int main() {
	int x = 0;
	{
		int x = 1;
	}
	int y = x + 1;
}
```

Quanto vale “y” alla fine? Perché

Esercizio 3 - Soluzione

Vale 1! Perché la seconda “x” vive solo all’interno del blocco in cui è definita

```
int main() {
	int x = 0;
	{
		int x = 1;
	}
	int y = x + 1;
}
```

L’istruzione scanf

```
scanf(“stringa di formato”, &var1, &var2…)
```

- Permette di leggere una stringa di testo da console ed interpretarla per estrarre dei dati.
- La struttura del testo di input viene specificata mediante una stringa di formato (primo argomento)
- Gli argomenti rimanenti sono le variabili in cui i dati devono essere inseriti

L’istruzione scanf

- Esempio:

```
scanf(“%d%d”, &x, &y)
```

- Legge due interi da standard input (la console) e li inserisce rispettivamente in “x” e “y”
- Il primo formato finisce nella prima variabile, e così via
- I nomi di variabili devono essere preceduti da “&” (più avanti nel corso si svelerà l’arcano…)

L’istruzione scanf

- Un altro esempio:

```
scanf(“%d%f”, &x, &y)
```

- Cosa fa?

L’istruzione scanf

- Un altro esempio:

```
scanf(“%d%f”, &x, &y)
```

- Cosa fa?
- Legge un intero ed un float, separati da spazi o “tab”
- Spazi e “tab” contano come separatori (non serve specificari)

L’istruzione scanf

- Ancora:

```
scanf(“%d;%f”, &x, &y)
```

- Legge un intero ed un float, che devono essere separati da un “;”
- Se la stringa di formato contiene altri caratteri oltre a quelli per le conversioni “%...”, questi devono comparire nella stringa di input
- Se mettete uno spazio, questo deve comparire nella stringa di input

L'istruzione scanf

- Quali conversioni?
  - "%d": intero
  - "%f": float
  - "%c": un carattere
  - "%ld": long int , mentre "%lf": double
- Speciale:
  - "%*d", "%*c", …: legge un dato e poi lo scarta
  - Es. `scanf("%*d%d", &x)`
  - Legge due interi e mette il secondo in "x"

Esercizio 0

```
#include <stdio.h>

int main() {
   int x;
   printf("Inserire la stringa di input: ");
   scanf("%d", &x);
   printf("Avete inserito: %d\n", x);
   return 0;
}
```

Esercizio 1

Modificate il programma precedente in modo che legga:

1. due interi, separati da spazi o “tab”
2. due interi, separati da “,”
3. una data, nel formato:
   “giorno/mese/anno”

In tutti i casi, stampate poi a video quello che avete letto!

Esercizio 1 - Soluzione

Modificate il programma precedente in modo che legga:

1. due interi, separati da spazi o “tab”:
   `scanf(“%d%d”, &x, &y);`
2. due interi, separati da “,”:
   `scanf(“%d,%d”, &x, &y);`
3. una data, nel formato “giorno/mese/anno”:
   `scanf(“%d/%d/&d”, &gg, &mm, &aa);`

Esercizio 2

Provate a togliere il “&” da una variabile.

Cosa succede?

Esercizio 2 - Soluzione

Provate a togliere il “&” da una variabile.

Cosa succede?

Un errore a run time! In particolare un segmentation fault. Questo succede perché il programma cerca di memorizzare il dato letto in una locazione di memoria su cui ha l’accesso in scrittura.

Esercizio 3

Modificate il programma precedente in modo che legga:

1. Un carattere
2. Due caratteri
3. Un intero ed un float, separati da qualunque carattere

Esercizio 3 - Soluzione

Modificate il programma precedente in modo che legga:

1. Un carattere
   ```
   scanf("%c", &x, &y);
   ```
2. Due caratteri
   ```
   scanf("%c%c", &x, &y);
   ```
3. Un intero ed un float, separati da qualunque carattere
   ```
   scanf("%d%*c%f", &x, &y);
   ```

Una Trappola

Lo spazio bianco e' a tutti gli effetti un carattere…

quindi nella lettura di caratteri

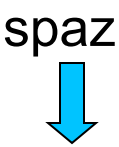
```
scanf("%d%c%c",&intero1,&car1,&car2);
printf("%d,%c,%c",intero1, car1,car2);
```

```
12 A B
12, ,A
```

La scanf ha preso lo spazio come se fosse il carattere inserito !

Una Soluzione

Usare un separatore (anche lo spazio stesso)




```
scanf("%d %c %c",&intero1,&car1,&car2);
printf("%d,%c,%c",intero1, car1,car2);
```

```
12 A B
12,A,B
```

Un’altra trappola

```
printf("Inserire un numero reale: ");
scanf("%f",&reale1);
printf("\nInserire un carattere: ");
scanf("%c",&car1);
printf("\nLetti: %f,%c",reale1, car1);
```

Questo frammento di programma sembra corretto...

Un’altra trappola

...ma il risultato e’ questo:

```
Inserire un numero reale:
12.4

Inserire un carattere:
Letti: 12.400000,
```

Motivo

L' I/O e’ bufferizzato: i caratteri letti da tastiera sono memorizzati in un buffer, incluso il fine linea.

Quando leggete il float, il fine linea rimane non consumato nel buffer di input.

Quindi quando poi andate a leggere un carattere, viene letto il fine linea che era rimasto nel buffer!

Una soluzione

Leggere il carattere "spurio""

```
printf("Inserire un numero reale: ");
scanf("%f",&reale1);
scanf("%*c"); /* letto e buttato via */
printf("\nInserire un carattere: ");
scanf("%c",&car1);
printf("\nLetti: %f,%c",reale1, car1);
```

Una Seconda Soluzione

Vuotare il buffer:

```
char buffer[DIM];
…
printf("Inserire un numero reale: ");
scanf("%f",&reale1);
gets(buffer); /* Si vuota il buffer */
printf("\nInserire un carattere: ");
scanf("%c",&car1);
printf("\nLetti: %f,%c",reale1, car1);
```

Precisazione

Questo problema si verifica solo con la lettura di caratteri.

Negli altri casi il doppio carattere nel buffer e' considerato come sequenza di separatori e scartato.