



Insegnamento di Programmazione (6 CFU)

Introduzione al linguaggio C - continuazione

Ing. Nadia Ranaldo
Dipartimento di Ingegneria
Università degli Studi del Sannio

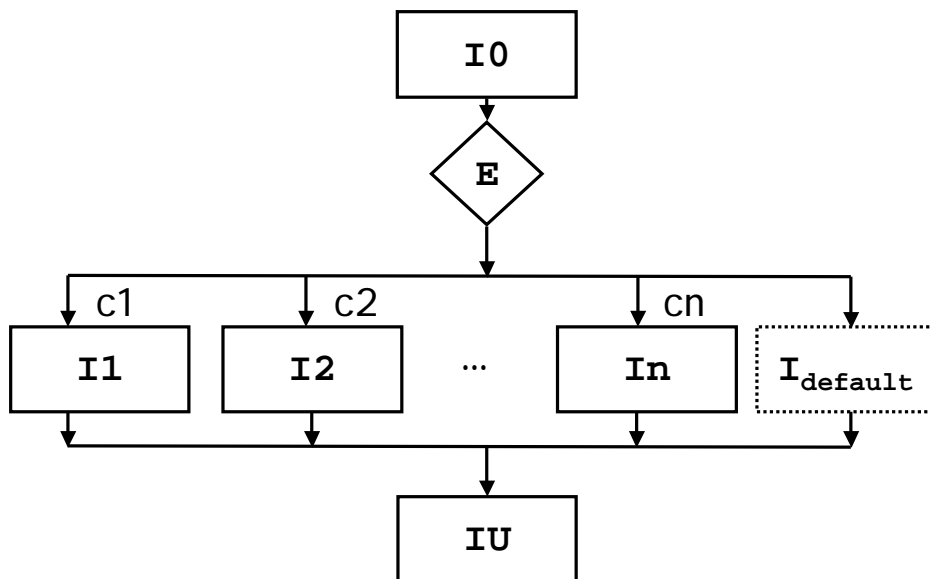
switch



- Struttura di scelta multipla che controlla se una espressione assume un valore all'interno di un certo insieme di costanti e si comporta di conseguenza

Attenzione: i valori ammessi come possibili scelte devono essere costanti!!!

switch



switch: sintassi



```
switch (espressione){  
    case costante1: sequenza_di_istruzione1;  
    break;  
    case costante2: sequenza_di_istruzione2;  
    break;  
    ...  
    case costanteN: sequenza_di_istruzioneN;  
    break;  
    default: sequenza_di_istruzioneDefault;  
    break;  
}
```



switch: semantica

break provoca l'uscita immediata dallo switch

Valuta **espressione**

se ha valore **costante1** esegui **sequenza_di_istruzione1**; break

se ha valore **costante2** esegui **sequenza_di_istruzione2**; break

...

se ha valore **costanteN** esegui **sequenza_di_istruzioneN**; break

altrimenti (se il valore di **espressione** è diverso da **costante1** ,..., **costanteN**) esegui **sequenza_di_istruzioneDefault**; break (opzionale)



Esempio



```
#include<stdio.h>
main() {
  int day;
  scanf("%d", &day);
  switch (day) {
    case 0: printf("Sunday\n"); break;
    case 6: printf("Saturday\n"); break;
    default: printf("Weekday\n"); break;
  }
}
```

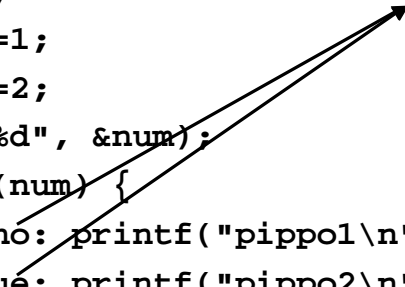


Esempio



```
#include<stdio.h>
main()
{int num;
  int uno=1;
  int due=2;
  scanf("%d", &num);
  switch (num) {
    case uno: printf("pippo1\n"); break;
    case due: printf("pippo2\n"); break;
    default: printf("pippo_default\n");
  }
}
```

SBAGLIATO
Valori non costanti



Osservazione



- **break** non è strettamente indispensabile.
- Se non è presente viene eseguita sequenzialmente ogni istruzione a partire dal **case** che è stato raggiunto



Esempio senza break



```
#include<stdio.h>
main() {
  int day;
  scanf("%d", &day);
  switch (day) {
    case 0: printf("Sunday\n");
    case 6: printf("Saturday\n");
    default: printf("Weekday\n");
  }
}
```



Un altro esempio



Possono esserci più etichette per una stessa sequenza di istruzioni

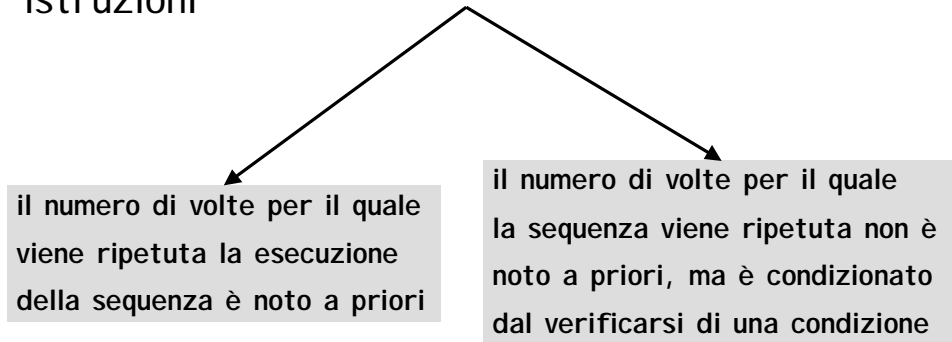
```
#include <stdio.h>
main()
{char car;
  scanf("%c", &car);
  switch (car) {
    case 'a': case 'e': case 'i':
    case 'o': case 'u':
      printf("Vocale minuscola\n"); break;
    case 'A': case 'E': case 'I':
    case 'O': case 'U':
      printf("Vocale maiuscola\n"); break;
    default: printf("Non e' vocale\n"); break;
  }
}
```



Istruzioni cicliche



- Consentono di realizzare cicli di elaborazione, ossia l'esecuzione ripetuta di una sequenza di istruzioni



Esercizio proposto



- Scrivere un programma che visualizza a video una serie di numeri ordinali in inglese (1st, 2nd, 3rd, 4th, 5th) compresi tra due limiti specificati dall'utente

Ad esempio:

```
Inserire il limite inferiore: 2
Inserire il limite superiore: 4
2nd
3rd
4th
```

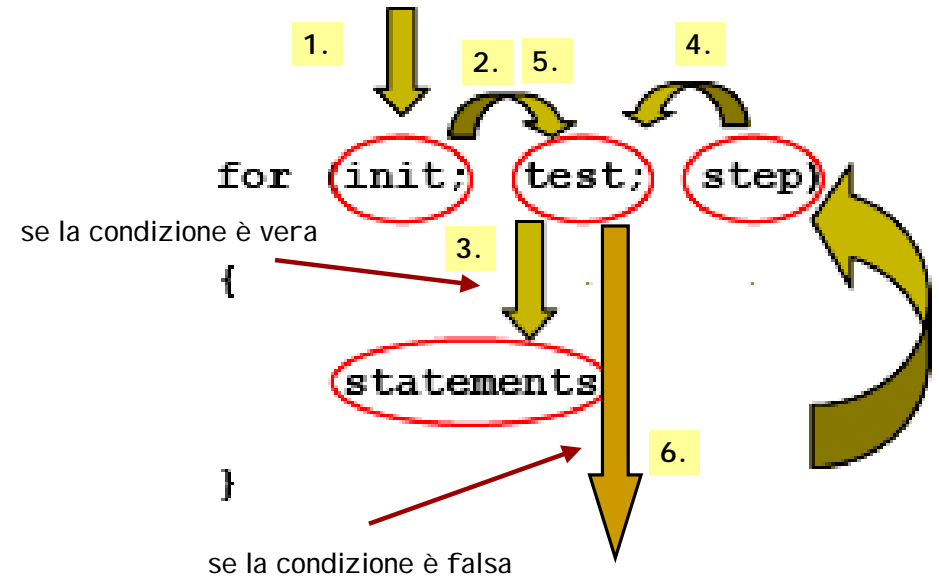
Istruzioni cicliche

❖ for

❖ while

❖ do-while

Il ciclo for



Esempio

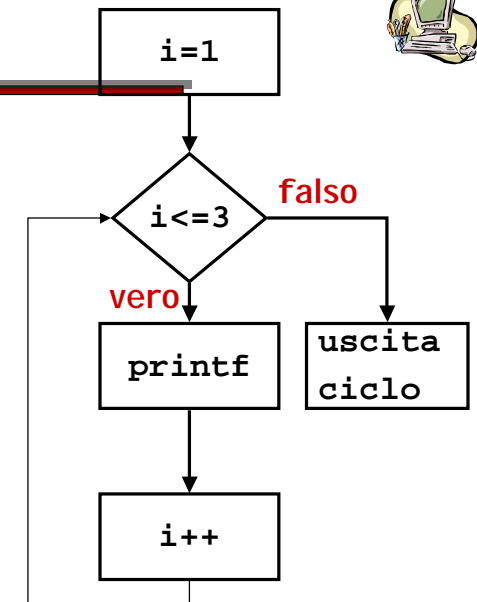
- Vogliamo visualizzare tre volte la scritta
- Ciao, mondo. il numero di volte per il quale viene ripetuta l'esecuzione della sequenza è noto a priori: 3

```
...
printf("Ciao, mondo.\n");
printf("Ciao, mondo.\n");
printf("Ciao, mondo.\n");
...
```

Usando il for

```
#include<stdio.h>
main()
{
    int i;
    for (i=1; i<=3; i++)
        printf("Ciao, mondo.\n");
}
```

Variabile contatore





Sintassi e Semantica del for



- for (**inizializzazione**; **condizione**; **incremento**) **istruzione**
- **inizializzazione**
 - Viene eseguita una volta sola
 - Serve per dare un valore iniziale alle variabili contatore
 - Può anche non essere presente, ed allora dopo la parentesi tonda aperta viene subito il punto e virgola



Semantica (cont.)



for (**inizializzazione**; **condizione**; **incremento**) **istruzione**

- **condizione**
 - Viene valutata ogni volta prima di eseguire le istruzioni del ciclo... (è la domanda!!!)
 - Se **condizione** è vera si esegue ancora **istruzione**
 - Se **condizione** è falsa si esce dal ciclo **for** passando all'istruzione successiva del programma
- Può anche non essere presente, ed allora lo spazio tra i due punti e virgola rimane vuoto. Il compilatore valuta vera la condizione assente, quindi continua ad eseguire **istruzione** → realizza un *loop infinito*



Semantica (cont.)



for (**inizializzazione**; **condizione**; **incremento**) **istruzione**

- **incremento**
 - Viene eseguita alla fine di ogni ciclo
 - Modificano ad esempio le variabili contatore, ossia la variabile usata per contare (incremento o decremento)
 - Può anche non essere presente, ed allora dopo il secondo punto e virgola viene subito la parentesi tonda chiusa

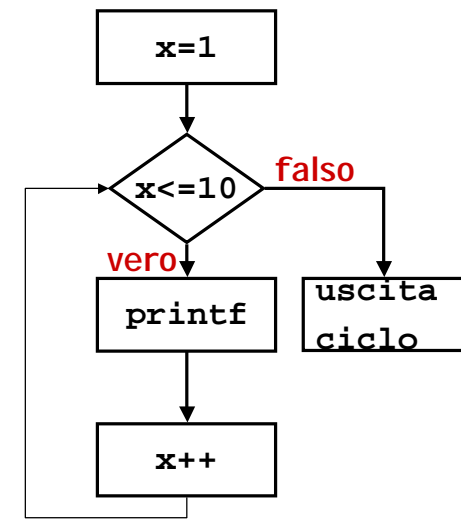


```

{
int x;
for (x=1; x<=10; x++) printf("%d\n",x);
}

```

1
2
3
4
5
6
7
8
9
10



x	x<=10?
1	si
2	si
3	si
4	si
5	si
6	si
7	si
8	si
9	si
10	si
11	no



Esempio: loop infinito



```

{
int x;
for ( ; ; ) printf("loop");
}
{
int x;
for ( x=0; ; ) printf("loop");
}
{
int x;
for ( x=0; x==0; ) printf("loop");
}

```

looplooplooplooplooploop
looplooplooplooplooploop
looplooplooplooplooploop
looplooplooplooplooploop
looplooplooplooplooploop
looplooplooplooplooploop
looplooplooplooplooploop

Compilatore non segnala errore!



Osservazioni



- Poiché la condizione viene valutata prima di ogni ciclo, il **for** permette anche di non eseguire nemmeno una volta il ciclo

Esempio

```

{
int i;
for (i=10; i<7; i+=2)
    printf("%d\n", i);
}

```

Non stampa nulla



Osservazioni (cont.)



- **Inizializzazione** ed **incremento**, nella sintassi del **for**, possono contenere più istruzioni, che dovranno essere separate da virgola.

Esempio

```

{
int x, y;
for (x=0, y=0; x+y<10; x++, y+=3)
    printf("%d\n", x+y);
}

```



Esempio di uso del for



Visualizzare i numeri da 100 a 0 a passo di 2

```

#include <stdio.h>
main() {
    int i;
    for (i=100; i>=0; i-=2)
        printf("%d", i);
}

```

- 100
- 98
- 96
- 94
- 92
- 90
- ...
- 2
- 0



Esempio di uso del `for`



Calcolare la somma di 5 numeri interi immessi dall'utente

```
#include <stdio.h>
main() {
    int i, somma, numero;
    somma=0;
    for (i=1; i<=5; i++){
        printf("Dammi il numero: ");
        scanf("%d", &numero);
        somma=somma + numero;
    }
    printf("Somma = %d\n", somma);
}
```

Variabile accumulatore:
 Contiene di volta in volta la somma dei numeri immessi fino a quel momento

25



Esempio (cont.)



Utile far apparire il numero d'ordine d'inserimento:

```
#include <stdio.h>
main() {
    int i, somma, numero;
    somma=0;
    for (i=1; i<=5; i++) {
        printf("Dammi il numero %d: ", i);
        scanf("%d", &numero);
        somma=somma + numero;
    }
    printf("Somma = %d\n", somma);
}
```

26



Quando usare `for`?



- Per realizzare cicli basati su una variabile che si incrementa e diminuisce di una certa quantità ogni volta che il ciclo viene eseguito (**variabile contatore**)
- Quando la variabile raggiunge un determinato valore, l'esecuzione del ciclo viene interrotta

27



Esercizio



- Visualizzare una tabella in cui una riga contiene un numero, il quadrato e il cubo. Utilizzare i numeri da 1 a 25.

1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
•	•	•
•	•	•

28



Soluzione 1



```
int i;
for(i=1;i<=100;i++)
    printf("%d  %d  %d\n", i, i*i, i*i*i);
```

1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000
11	121	1331

due spazi tra un numero ed un altro ->

Una tabella di numeri non allineati verticalmente!

-i numeri occupano lo spazio strettamente necessario (che aumenta man mano che il numero diventa più grande)

-Soluzione?

29



Soluzione 2



```
int i;
for(i=1;i<=100;i++)
    printf("%3d %5d %7d\n",i,i*i,i*i*i);
```

1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000
11	121	1331

Fissare lo spazio occupato di ogni numero che verrà visualizzato

30



Esercizi



1. Scrivere un programma che richiede all'utente un numero naturale n e calcola la somma dei primi n numeri naturali.
2. Scrivere un programma che richiede all'utente un numero naturale n e calcola la somma di n numeri inseriti dall'utente.
3. Scrivere un programma che richiede all'utente un numero naturale n e calcola il massimo degli n numeri inseriti dall'utente.

31



Esercizi (cont.)



4. Scrivere un programma che richiede all'utente un numero naturale n e calcola il fattoriale di n , indicato con $n!$

$$0! = 1$$

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot 2 \cdot 1 \quad n > 0$$

Esempio:

$$4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$$

$$6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720$$

32



Analisi dell'esercizio 1



- Comprensione del problema:

Input n

Output somma = 1+..+n

Esempio

Input n = 5

Output somma= 1+2+3+4+5=15

- Algoritmo:

- Leggere il numero n da tastiera
- Sommare man mano i valori utilizzando una variabile accumulatore somma inizializzata a 0
- Visualizza somma

33



Somma dei primi n numeri (Es. 1)



```
#include <stdio.h>
main() {
    int n, somma, i;
    printf("Inserisci n: ");
    scanf("%d", &n);
    somma=0;
    for (i=1; i<=n; i++)
        somma=somma+i;
    printf("La somma dei primi n num = %d", somma);
}
```

34



Analisi dell'esercizio 2



- Comprensione del problema:

Input n e n numeri immessi da tastiera num_i con i:1..n

Output somma = num₁+num₂+..+num_n

Esempio:

Input n = 3 num1=2 num2=3 num3=6

Output somma= 2+3+6=11

- Algoritmo:

- Leggere il numero n da tastiera
- Sommare man mano i valori letti da tastiera utilizzando una variabile accumulatore somma inizializzata a 0
- Visualizza somma

35



Somma di n numeri (Es. 2)



```
#include <stdio.h>
main()
{ int n, somma=0, num, i;
  printf("Inserisci n: ");
  scanf("%d", &n);
  printf("Ora inizia l'inserimento di %d numeri\n", n);
  for (i=1; i<=n; i++) {
      printf("Inserisci il numero %d: ", i);
      scanf("%d", &num);
      somma=somma+num;
  }
  printf("La somma = %d", somma);
}
```

36



Analisi dell'esercizio 3



- Comprensione del problema:

Input n e n numeri immessi da tastiera num_i con $i:1..n$

Output $max = \max(num_1, num_2, \dots, num_n)$

Esempio:

Input n = 3 num1=2 num2=3 num3=6

Output max= 6

Algoritmo:

- Leggere il numero n da tastiera
- Leggere il primo numero e assegnarlo alla variabile intera max che verrà utilizzata in seguito per contenere man mano il numero massimo tra quelli letti fino a quel momento
- Leggere gli altri numeri e man mano calcolare il massimo eseguendo un confronto tra il numero letto num e la variabile max
- Se $max < num$ aggiorna max a num, altrimenti non fa nulla
- Visualizza max

37



Esercizio 3: Massimo di n numeri



```
#include <stdio.h>
main(){
    int n, max, num, i;
    printf("Inserisci n: ");
    scanf("%d", &n);
    printf("Inserisci il numero 1: ");
    scanf("%d", &num);
    max=num;
    for (i=2; i<=n; i++) {
        printf("Inserisci il numero %d: ", i);
        scanf("%d", &num);
        if (max < num)
            max=num;
    }
    printf("Il massimo= %d", max);
}
```

38



Analisi dell'esercizio 4



- Comprensione del problema:

Input n

Output fattoriale = $1*2*..*n$

Esempio:

Input n = 3

Output fattoriale= $3*2*1=6$

Algoritmo:

- Leggere il numero n da tastiera
- Inizializzare la variabile intera fattoriale a 1 che verrà utilizzata in seguito per contenere man mano il valore parziale del fattoriale
- Moltiplicare fattoriale per n, che corrisponde al numero di partenza e poi man mano per i valori ottenuti decrementandolo n di uno fino ad arrivare ad 1 (utilizzare una variabile i di appoggio per contenere i valori da n fino a 1)
- Visualizza fattoriale

39



Esercizio 4: Fattoriale di n



```
#include <stdio.h>
main() {
    int n, fatt, i;
    printf("Inserisci n: ");
    scanf("%d", &n);
    fatt=1;
    for (i=n; i>=1; i--)
        fatt=fatt*i;
    printf("Il fattoriale di n = %d", fatt);
}
```

40

while

- Permette di ottenere la ripetizione ciclica di una istruzione sotto il controllo di una condizione di terminazione

- Sintassi

while (espressione) istruzione

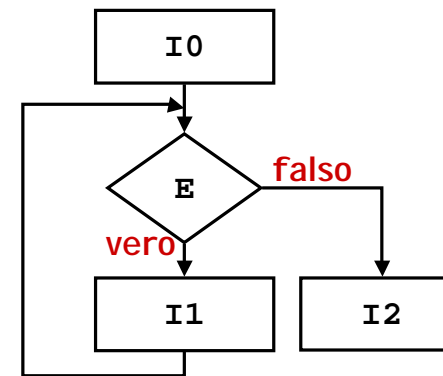
- Semantica

- **espressione** deve essere valutata ogni volta prima di eseguire **istruzione**

- se la valutazione di **espressione** è vera allora **istruzione** viene eseguita, altrimenti no, e si esce dal ciclo

41

while: diagramma di flusso



```
I0;  
while (E) I1;  
I2;
```

42

Analisi

- Poiché la valutazione dell'espressione è effettuata all'inizio, il ciclo stesso può essere eseguito:

- ❖ zero volte
- ❖ un numero finito di volte
- ❖ infinite volte

43

Esempio

```
int x=12;  
int y= 1;  
while(y<x)  
    y=y*2;
```

- Quanto vale **y** alla fine dell'esecuzione

Y=16



44

Soluzione

```
int x=12;
int y=1;
while(y<x)
    y=y*2;
```

y	x
1	12
2	
4	
8	
16	

45

Errori frequenti: ripetizioni infinite

Esempi

```
...
while (x > 0)
a = b + c;
```

poiché il valore di **x** non viene in alcun caso modificato.....

```
...
while(i > n) {
...
i = i + 1; }
```

i varia, ma aumenta

46

Esempio

```
#include<stdio.h>
main() {
int x=0;
while (x==0)
    printf("loop");
}
```

loop infinito

```
looplooplooplooplooploop
looplooplooplooplooploop
looplooplooplooplooploop
looplooplooplooplooploop
looplooplooplooplooploop
looplooplooplooplooploop
looplooplooplooplooploop
```

47

Convergenza del ciclo

- Assicurarsi, in prima istanza, che nel corpo del **while** venga alterato il valore di almeno una delle informazioni coinvolte in **E**
- Assicurarsi, in ogni caso, che in un numero finito di iterazioni, **E** diventi falsa
- Assicurarsi che a tutte le informazioni usate nell'espressione **E** sia stato assegnato un valore prima del **while**

48

Esempio

- Somma numeri interi forniti in ingresso dalla tastiera finché non viene dato il valore zero

49

Soluzione

```
#include<stdio.h>
main() {
    int num=1;
    int somma=0;
    while (num !=0) {
        printf("Dammi un numero: ");
        scanf("%d", &num);
        somma=somma+num;
    }
    printf("La somma = %d%", somma);
}
```

Alla variabile `num` si è assegnato il valore 1 per far in modo che il ciclo venga eseguito almeno una volta, ovviamente qualunque numero diverso da zero va bene

50

Soluzione alternativa

Alla variabile `num` si è assegnato il primo numero inserito dall'utente fuori dal ciclo

```
#include<stdio.h>
main(){
    int num;
    int somma=0;
    printf("Dammi un numero: ");
    scanf("%d", &num);
    while (num !=0){
        somma=somma+num;
        printf("Dammi un numero: ");
        scanf("%d", &num);
    }
    printf("La somma = %d", somma);
}
```

51

Relazione tra `for` e `while`

```
for (iniz; cond; inc)
{
    istruzioni;
}
```

.... è equivalente a:

```
iniz;
while (cond)
{
    istruzioni;
    inc;
}
```

52

Esercizio



- Siano **i** e **x** variabili intere. Data l'istruzione:
`for(i=1; i<=x; i++)
printf("%d\n", i*x);`
- Scrivere un programma equivalente usando il **while**

```
i=1;  
while(i<=x) {  
    printf("%d\n", i*x);  
    i=i+1;  
}
```

53

do-while



- **for** e **while** controllano la condizione di terminazione all'inizio del ciclo
- **do-while** controlla la condizione al termine di ogni iterazione
- consente di eseguire un ciclo condizionale da 1 ad infinite volte

54

do-while



- Sintassi

```
do  
    istruzione  
while (espressione);
```

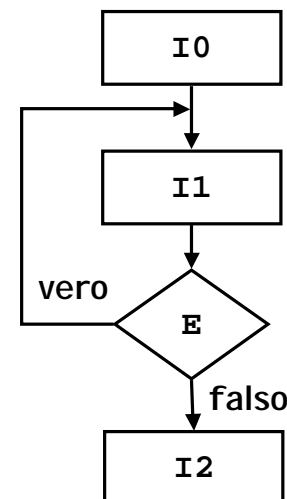
- Semantica

- **espressione** deve essere valutata ogni volta dopo aver eseguito **istruzione**

- se la valutazione di **espressione** è vera allora il ciclo viene ripetuto, cioè si riesegue **istruzione**, altrimenti si esce dal ciclo

55

do-while: diagramma di flusso



```
I0;  
do {  
    I1;  
} while (E);  
I2
```

56



Cosa stampa?



```
#include<stdio.h>
main()
{
  int n=12, k=5;
  do{
    n=n/2;
    k=k-1;
  } while(n>=k);

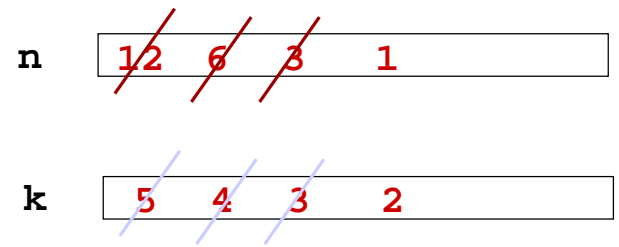
  printf("%d %d\n", n, k);
}
```

1 2



Perché?

```
int n=12, k=5;
do{
  n=n/2;
  k=k-1;}
while(n>=k);
printf("%d %d\n", n, k);
```



Esempio



```
int num;

do {
  scanf("%d", & num);
} while (num > 10);
```

Legge numeri finché non viene inserito un numero <=10



Riprendiamo l'esempio



- Somma di numeri interi forniti in ingresso dalla tastiera finché non viene dato lo zero



Soluzione con do-while



```
#include<stdio.h>
main()
{int num;
 int somma=0;
 do {
  printf("Dammi un numero: ");
  scanf("%d", &num);
  somma=somma+num;
 } while (num !=0);

 printf("La somma = %d", somma);
 }
```

61



Relazione tra do-while e while



```
do {
  istruzioni;
} while (espressione);
```

.... è equivalente a:

```
istruzioni;
while (espressione){
  istruzioni;
}
```

62



Esercizio



- Scrivere un programma che legge un carattere ed aspetta che l'utente digiti **s** oppure **n**. In questo caso stampa **OK**.

63



Problema???



```
#include<stdio.h>
main() {
  char car;
  do {
    printf("rispondi s oppure n: \n");
    scanf("%c", &car);
  } while ((car!='s') && (car!='n'));
  printf("OK");
}
```

64



Immissione di più caratteri



Se il programma richiede l'immissione di più valori in tempi diversi, l'inserimento di un carattere potrebbe costituire un problema, dato che la digitazione del tasto di **Invio** da tastiera corrisponde a un carattere accettato da

```
scanf("%c", ..)
```

In tal caso verrà utilizzata un'opportuna ulteriore lettura di un carattere in una variabile ausiliaria tramite un'istruzione del tipo

```
scanf("%c", &pausa);
```

5



Soluzione



```
#include<stdio.h>
main()
{
    char car, pausa;
    do {
        printf("rispondi s oppure n: \n");
        scanf("%c", &car);
        scanf("%c", &pausa);
    } while ((car!='s') && (car!='n'));
    printf("OK");
}
```

66



Suggerimenti



- Evitare programmi prolissi
- Non usare variabili più del necessario
- Memorizzare i risultati di computazioni intermedie quando tali risultati sono riusati

67



Esempio



- Programma che calcola il rapporto tra la somma e la differenza di due variabili e lo memorizza in **r**

prolisso	semplice
<code>somma = m+n;</code>	
<code>diff = m-n;</code>	<code>r = (m+n)/(m-n)</code>
<code>r = somma/diff</code>	

può essere utile per maggiore comprensione

68

Esempio



- Se un'espressione compare in più punti è meglio memorizzarla in una variabile piuttosto che ricalcolarla

```
s = (m+n)+5*(m+n)*(m+n);   sum = m+n;
t = (m+n)/(m-n);           s = sum+5*sum*sum;
...                          t = sum/(m-n);
u = 3*(m+n);                ...
                             u = 3*sum;
```

69

Esempio



```
while ((i<m-n+1) && (ris<max))
    ris=ris*i;
```

- **m** e **n** non vengono alterati nel corpo del ciclo.



```
sup = m-n+1;
while ((i<sup) && (ris<max))
    ris=ris*i;
```

Evita di ricalcolare
sempre l'espressione
m-n+1

70

Numero magico



- Generare un numero a caso e chiedere all'utente un numero fino a quando non è uguale a quello generato casualmente.

71

Numeri casuali



- **rand()** definita in **stdlib.h**
- per la generazione di numeri pseudo-casuali nell'intervallo compreso tra 0 e **RAND_MAX** (costante definita in **stdlib.h**)

72

Soluzione



```
#include <stdio.h>
#include <stdlib.h>
main()
{ int guess;
  int magic;
  magic = rand();
  do {
    printf("Indovina il numero magico: ");
    scanf("%d", &guess);
    if (guess==magic)
      printf("Bravo!!\n");
    else printf("Sbagliato. Riprova.\n");
  } while (magic != guess);
}
```

genera un
numero casuale

73

...così potrai
indivinare più
facilmente!

```
#include <stdio.h>
#include <stdlib.h>
main()
{ int guess;
  int magic;
  magic = rand() % 10;
  do {
    printf("Indovina il numero magico: ");
    scanf("%d", &guess);
    if (guess==magic)
      printf("Bravo!!\n");
    else printf("Sbagliato. Riprova.\n");
  } while (magic != guess);
}
```

74

Un idioma di programmazione: leggi-fino a-sentinella



- Condizione per l'uscita scelta dall'utente
- Il ciclo while deve ciclare all'infinito:

```
#define TRUE 1
while (TRUE) {
  .....
}
```
- In caso di uscita dal ciclo si usa l'istruzione break:

```
if (.....) break;
```

75

La somma con leggi-fino a-sentinella e while



```
/* sentinella.c: somma con sentinella
*/
#include <stdio.h>
#define TRUE 1
main()
{
  int i=0, appoggio, somma=0;

  while (TRUE) {
    printf("Inserire intero n.%d:", ++i);
    scanf ("%d", &appoggio);
    if ( appoggio == 0 ) break;
    somma += appoggio;
  }
  printf ("Somma: %d\n", somma);
}
```

- Definizione di TRUE come costante > 0
- Il ciclo infinito.
- Uso del contatore con incremento (prefisso!)
- La sentinella
 - se appoggio è 0 allora esci dal ciclo
 - Altrimenti continua la somma
- Output del risultato

76



Un errore tipico con il while



```

/* errore.c:
inserisce un numero diverso da 1
e poi fa qualche cosa..
*/
#include <stdio.h>
main()
{
  int appoggio=1;
  while (appoggio = 1) {
    printf("Digita intero diverso da 1:");
    scanf ("%d", &appoggio);
  }
  .....
}

```

l'idea:

- ciclare fino a quando l'utente immette 1
- il programma invece cicla all'infinito

Dov'è l'errore?

77



Un errore tipico con il while



```

/* errore.c:
inserisce un numero diverso da 1
e poi fa qualche cosa..
*/
#include <stdio.h>

main()
{
  int appoggio=1;

  while (appoggio = 1) {
    printf("Digita intero diverso da 1:");
    scanf ("%d", &appoggio);
  }
  .....
}

```

Confondere assegnazione con test di uguaglianza nei cicli

ECCOLO!

78



La somma con leggi-fino a-sentinella e for



```

/* sentinella2.c: sentinella con il for
*/
#include <stdio.h>
#define TRUE 1
main()
{
  int i=0, appoggio, somma=0;

  for ( ; TRUE ; ) {
    printf("Inserire intero n.%d:", ++i);
    scanf ("%d", &appoggio);
    if ( appoggio == 0 ) break;
    somma += appoggio;
  }
  printf ("Somma: %d\n", somma);
}

```

- Cambia solamente il ciclo
- Sono vuote
 - le condizioni di inizializzazione
 - e lo step
- Il break funziona anche con il ciclo for
- Commento: il for non serve solamente per cicli numerici

79



L'operatore condizionale (?:)



- Strettamente correlato con la struttura if-else
- Unico operatore ternario (accetta tre operandi)
- L'espressione risultante viene detta espressione condizionale
- **Sintassi:**
espressione logica? espressione1: espressione2
- **Semantica:**
 - Valuta **espressione logica** (ovvero una condizione)
 - Se è vera l'espressione condizionale assumerà il valore dato dalla valutazione **espressione**
 - Altrimenti l'espressione condizionale assumerà il valore dato dalla valutazione di **espressione2**

80



L'operatore condizionale (?:) (cont.)



Esempio:

```
int a,b,c;
scanf("%d\n",&a);
scanf("%d\n",&b);
c=(a>b)?a:b; /*c = max(a,b)*/
```

L'operatore condizionale permette di scrivere in maniera abbreviata:

```
if(a>b)
    c=a;
else c=b;
```

81



Esercizio



- Scrivere un programma C che prende in input un intero positivo e stampa la somma delle sue cifre (compreso gli addendi)
- Esempio:
 - se l'utente digita 215 il programma stamperà
- $5 + 1 + 2 = 8$
 - se l'utente digita 1765, il programma stamperà
- $5 + 6 + 7 + 1 = 19$

82



Esercizio



• Algoritmo

- Occorre stampare le cifre seguite dal segno + man mano che viene calcolata e sommata con le precedenti
- Quando si arriva all'ultima cifra occorre stampare la cifra seguita dal segno =
- La cifra può essere calcolata come il resto della divisione per dieci. Ad esempio $215\%10 = 5$
- Per procedere con la cifra successiva, occorre utilizzare il quoziente della divisione per 10. Ad esempio $215/10=21$ fino ad arrivare all'ultima cifra, ovvero fino a che la divisione per 10 non dà resto 0. Ad esempio $2\%10=2$ e $2/10=0$, quindi mi devo fermare!

83



Soluzione



```
#include<stdio.h>
main() {
    int n,s=0;
    printf("Ins.: \n");
    scanf("%d", &n);
    while(n>0){
        s+=n%10;
        printf("%d %c ", n%10,
              (n>=10)?'+':'=');
        n/=10;
    }
    printf("%d\n",s);
}
```

- Input di numero
- Ciclo while
 - estrae la cifra più a destra
 - la stampa con il '+' oppure '=' a seconda se è l'ultima cifra oppure no
 - divide per 10 numero per la prossima cifra.
- Output somma

84



Un esempio di uso del *for*



- Scrivere un programma C che stampa la tavola pitagorica 10 x 10 delle moltiplicazioni
- Da notare che:
 - dobbiamo stampare una tabella con due indici
 - per ogni valore di un indice riga, dobbiamo stampare tutte le colonne (per i valori delle colonne)
- Soluzione: si usano due *for* innestati
 - il primo serve a far variare l'indice di riga
 - il secondo (interno) fa variare l'indice di colonna

85



Un esempio di uso del *for*



```

/* tabella.c
 * Stampa la tavola pitagorica 10 x 10
 */
#define INIZIO 1
#define FINE 10
main()
{
  int riga, col;

  for (riga = INIZIO; riga <= FINE; riga++){
    for (col = INIZIO; col <=FINE; col++)
      printf ("%3d ", riga*col);
    printf ("\n");
  }
}

```

- Costanti simboliche
- Ciclo *for* esterno:
 - varia la riga
- ciclo *for* interno:
 - stampa tutte le colonne
 - a fine riga vai a capo

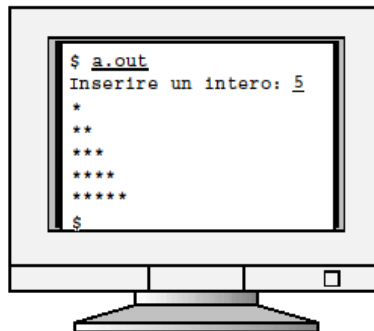
86



Esercizio proposto



- Scrivere un programma C che prende in input un intero *lung* e stampa a schermo righe di asterischi di lunghezza crescente da 1 a *lung*.



87