TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Macroarea di Ingegneria
Dipartimento di Ingegneria Civile e Ingegneria Informatica

Linguaggio C: Controllo

Valeria Cardellini

Corso di Calcolatori Elettronici
A.A. 2019/20

Strutture di controllo

- Sequenza
- Selezione: if e if-else
- Selezione multipla: switch
- Iterazione: while
- Iterazione: do…while
- Iterazione: for

Selezione: if-else

- La struttura di selezione if-else permette di eseguire in modo condizionale un blocco di codice in base alla verità di una condizione
- Sintassi

  if (condizione) { blocco-1 }
  else { blocco-2 }

- La parte else è opzionale
- Semantica
  - Viene valutata la condizione
  - Se la condizione è vera si esegue il blocco 1 di istruzioni
  - Se la condizione è falsa si esegue il blocco 2 di istruzioni (se presente)

```
blocco ::= {
  [ dichiarazioni e definizioni ]
  istruzione-1
  ...
  istruzione-n
}
```

Codice sorgente: pari_dispari.c

Errore subdolo

- Attenzione a non usare erroneamente nella condizione l'operatore di assegnamento (=) anziché l'operatore uguale a (==) per effettuare un confronto
- Es.:

```
int a=0;
if (a=0) printf("ERRORE\n");  /* printf mai eseguita */
```

if-else annidati

- If-else possono essere annidati
- In assenza di parentesi, `else` si riferisce sempre a `if` più vicino
  - Es:

```
if (a > 0) if (b > 0) printf("b  positivo\n");
else  printf("?\n");
```

- Per evitare ambiguità di significato, usare le parentesi
  - Es:

```
if (a > 0) {
   if (b > 0) printf("b  positivo\n");
   else  printf("?\n");
}
```


if-else: valutazione di condizioni complesse

- Nelle condizioni contenenti gli operatori booleani && e ||, le sotto-espressioni vengono valutate da sinistra a destra
  - Non appena la verità della condizione è determinata, la valutazione delle sotto-espressioni è sospesa
  - In particolare:
    - ✓ nel valutare `(e1 && e2)` se la valutazione di `e1` restituisce `0`, allora `e2` non è valutata
    - ✓ nel valutare `(e1 || e2)` se la valutazione di `e1` restituisce `1`, allora `e2` non è valutata
- Fare attenzione ad eventuali side-effect

Operatore condizionale

- Il C prevede espressioni condizionali
  - Il valore dell'espressione dipende dal valore di una condizione booleana
  - Le espressioni condizionali usano l'operatore condizionale (terziario) **?**
- Sintassi

  `condizione?espressione1: espressione2`
  - condizione è una condizione booleana
  - espressione1 e espressione2 sono espressioni dello stesso tipo

Operatore condizionale

- Semantica
  - Ordine di valutazione da sinistra a destra
    - ✓ Viene prima valutata la condizione
    - ✓ Se la condizione è vera, viene valutata espressione1
    - ✓ Altrimenti, viene valutata espressione2
- Esempi

```
printf("massimo = %d\n", (a > b) ? a : b);

int x=3, y=0;
int z = x==y ? x : y;            // z=y=0, x=3

int x=3, y=0;
int z = y<=x ? y++ : 1;          // z=0, y=1, x=3
```

Operatore condizionale

- Se uno dei due rami non viene scelto, l'espressione corrispondente non viene valutata
  - Attenzione ad eventuali side-effect
- Esempio:

```
int x=3, y=0;
int z = x!=y? ++y : ++x;      // z=y=1, x=3
```


Selezione multipla: switch

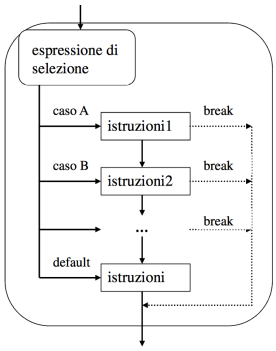
- Consente di realizzare una selezione a più vie
- Sintassi

```
switch (selettore) {
    case etichetta1: istruzioni-1
    break;
    …
    case etichettaN: istruzioni-N
    break;
    default: blocco-default
}
```

  - Selettore: di tipo intero o char
  - Etichetta: espressione intera (o carattere) di valore costante
    - ✓ Solo letterali interi (o caratteri) o costanti inizializzate con espressioni costanti

Selezione multipla: switch

- I vari rami dello switch non sono necessariamente mutuamente esclusivi
  - Imboccato un ramo, si eseguono anche tutti i rami successivi a meno che non ci sia il comando break a forzare esplicitamente l'uscita



Iterazioni in C

- Il C offre tre costrutti di iterazione
  - while
  - for
  - do-while
- while è il costrutto più generale e consente di esprimere qualsiasi ciclo esprimibile con for e do-while
  - In alcune situazioni, gli altri costrutti possono migliorare soprattutto leggibilità e chiarezza del codice

Iterazione: while

- Il costrutto di iterazione while consente la ripetizione di un blocco di istruzioni, controllandola con una condizione
  - Come in Python
- Sintassi

  `while (condizione) { blocco }`
  - condizione è un'espressione
  - blocco (il *corpo del ciclo*) è un blocco di istruzioni
- Semantica
  - La condizione è valutata *prima* di ogni iterazione
  - Il blocco viene eseguito finchè la condizione è vera

Esempio: while

```
/* Legge e stampa caratteri */

#include <stdio.h>

int main(void) {
        int c;

        c = getchar();
        while (c != EOF) {
                putchar(c);
                c = getchar();
        }
        return 0;
}
```

Codice sorgente: readprint_char.c

Funzioni getchar e putchar

- `getchar`: legge il prossimo carattere dal canale di input (standard input o *stdin*: tastiera) e lo restituisce come valore intero

```
int getchar(void);
```

  - Il carattere letto è restituito come `int`
  - Se si raggiunge la fine del file viene letto il carattere EOF (End Of File)
    - ✓ Per chiudere il canale di input producendo EOF da tastiera: Ctrl-D in sistemi operativi Unix-like, Ctrl-Z in sistemi operativi Windows
      - Per verificare la combinazione

```
$> stty -a
```

  - In caso di EOF, viene riscontrato un errore in lettura e viene restituito il valore della costante simbolica di tipo `int EOF` (che generalmente vale -1) definita in stdio.h

Funzioni getchar e putchar

- putchar: copia sul canale di output (standard output o *stdout*: schermo) il carattere memorizzato in ch

```
int putchar(int ch);
```

  - Se non ci sono errori, restituisce il carattere stampato
  - Se si verifica un errore, restituisce EOF

Esempio: while

- Versione più compatta
  - Notare il side-effect (chiamata della funzione nella condizione)

```
#include  <stdio.h>

int  main(void)  {
    int  c;

    while  ((c  =  getchar())  !=  EOF)
        putchar(c);
    return  0;
}
```

Codice sorgente: readprint_char_v2.c

Errori comuni nei cicli while

- Mancata inizializzazione di una variabile che viene utilizzata nella condizione del ciclo
- Mancato aggiornamento delle variabili che compaiono nella condizione del ciclo
- Numero di iterazioni errato (in genere di una unità)

Iterazione: do…while

- Il costrutto di iterazione do…while consente la ripetizione di un blocco di istruzioni, controllandola con una condizione
  - A differenza di while, il blocco di istruzioni viene eseguito almeno una volta
- Sintassi

  ```
  do { blocco } while (condizione);
  ```
  - blocco (il corpo del ciclo) è un blocco di istruzioni
  - condizione è un'espressione
- Semantica
  - La condizione è valutata *dopo* ogni iterazione del ciclo
  - Il blocco viene eseguito *almeno una volta* e finché la condizione è vera

Equivalenza tra while e do…while

```
do {
   printf("Inserisci un intero > 0:");
   scanf("%d", &i);
} while (i <= 0);
```

Equivale a:

```
printf("Inserisci un intero > 0:");
scanf("%d", &i);
while (i <= 0) {
   printf("Digita un intero > 0:");
   scanf("%d", &i);
}
```

Iterazione: for

- Sintassi

```
for (espr-1; espr-2; espr-3)
{ blocco }
```

  - espr-1: inizializza la variabile di controllo
  - espr-2: condizione di terminazione
  - espr-3: modifica della variabile di controllo
  - blocco: corpo del ciclo
- Semantica: è equivalente a

```
espr-1;
while (espr-2) {
    blocco
    espr-3;
}
```


for: osservazioni

- Ciascuna delle tre parti del for può mancare
  - In ogni caso i ; vanno comunque inseriti
  - Se manca la condizione, viene assunta vera
  - Es. ciclo infinito:

```
for (; ;)
```

    equivale a

```
while (1)
```

- Nel caso generale, inizializzazione ed incremento della variabile di controllo possono anche essere una sequenza di espressioni con side-effect separate da ,
  - Questo consente di inizializzare e/o incrementare più variabili contemporaneamente
- Possono esserci più variabili di controllo

Analizziamo le strutture di controllo tramite esempi

- `while`: contatore_while.c
- `do...while`: contatore_do_while.c
- `for`: contatore_for.c e interesse_composto.c
- `while` e `if-else`: risultati_esami.c
- `while` e `switch`: conta_voti.c

Istruzioni di salto

- Le istruzioni di salto consentono il trasferimento arbitrario del controllo di flusso
  - break: salta all'istruzione successiva al ciclo o allo switch in cui essa compare
  - continue: salta alla successiva iterazione del ciclo
  - goto: salta all'istruzione indicata tramite etichetta
- Anche l'istruzione `return` può essere usata nelle funzioni per modificare il flusso di esecuzione

Istruzione break

- Altera il flusso del controllo e permette di uscire dai cicli e da switch
- Quando viene usata nei cicli
  - si perde la strutturazione del programma
  - ma si guadagna in efficienza
- Esempio

```
int i, n;
for (i=1; i<10; i++) {
    scanf("%d", &n);
    if (n > 0) printf("%d", n);
    else break;
}
```


Istruzione continue

- L'istruzione `continue` si applica solo ai cicli e comporta il passaggio alla successiva iterazione del ciclo, saltando le eventuali istruzioni che seguono nel corpo del ciclo
- Esempio

```
/* Stampa soltanto i multipli di 3 */
int i;
for (i=1; i<n; i++) {
  if (i%3 != 0) continue;
  printf("%d\n", i);
}
```

Istruzione di salto goto

- L'istruzione di salto `goto` comporta un'interruzione del flusso dell'esecuzione del programma, che prosegue con l'istruzione specificata nel `goto`
- Deriva dal linguaggio macchina dove è l'unica struttura di controllo e ha un ruolo fondamentale per consentire la realizzazione dei cicli
  - Fino agli anni '70 anche tutti i linguaggi ad alto livello (es. Fortran) erano pesantemente basati sul goto
  - Il teorema di Böhm e Jacopini dimostra che tale istruzione non è necessaria ai fini della completezza del linguaggio

"Qualunque algoritmo può essere implementato utilizzando tre sole strutture di controllo: la sequenza, la selezione ed il ciclo (iterazione), da applicare ricorsivamente alla composizione di istruzioni elementari."

Istruzione di salto goto

- Sintassi

  `goto etichetta`
  - Etichetta è un identificatore che individua un'istruzione del programma; si usa la sintassi `etichetta: istruzione`
- Semantica
  - L'esecuzione del programma prosegue con l'istruzione specificata dall'etichetta

Istruzione di salto goto

- Esempio

```
int i=0;
int x;
while (i < 100) {
   printf("Inserisci un intero: \n");
   scanf("%d", &x);
   if(x < 0) goto errore;
   ...
   i++;
}
...
errore: printf("Termina a causa di errore
nell'input\n");
```