

Strutture di controllo

Programmazione strutturata

Ramificazione

Introduzione all'iterazione

Liceo G.B. Brocchi
Classi prime Scientifico - opzione scienze applicate
Bassano del Grappa, Gennaio 2023

Ancora sulla ramificazione

- Scriviamo un programma che si comporta in questo modo:
 - se l'utente del programma inserisce il carattere 'a' da stdin, il programma stamperà 'A'
 - se l'utente del programma inserisce il carattere 'b' da stdin, il programma stamperà 'B'
 - se l'utente del programma inserisce il carattere 'c' da stdin, il programma stamperà 'C'
 - se l'utente del programma inserisce il carattere 'c' da stdin, il programma stamperà 'D'
 - con qualsiasi altro input, il programma stamperà su stdout il messaggio *«input not accepted, please enter a, b, c, or d»*

Un programma logicamente sbagliato

```
int main() {
    char choice;
    printf("enter a character: ");
    scanf("%c", &choice);
    if (choice == 'a') {
        putchar('A');
        putchar('\n');
    }
    if (choice == 'b') {
        putchar('B');
        putchar('\n');
    }
    if (choice == 'c') {
        putchar('C');
        putchar('\n');
    }
    if (choice == 'd') {
        putchar('D');
        putchar('\n');
    }
    else {
        puts("input not accepted, please enter a, b, c, or d");
    }
}
```

**Cosa succede se l'utente
inserisce 'a', oppure 'b', oppure
'c', oppure 'd'?**

Un programma logicamente sbagliato

```
cyofanni@LAPTOP-I0S1RKRC:~/Desktop/high-school-cs-class/c_lectures/starter$ ./if_else_if
enter a character: a
A
input not accepted, please enter a, b, c, or d
cyofanni@LAPTOP-I0S1RKRC:~/Desktop/high-school-cs-class/c_lectures/starter$ ./if_else_if
enter a character: b
B
input not accepted, please enter a, b, c, or d
cyofanni@LAPTOP-I0S1RKRC:~/Desktop/high-school-cs-class/c_lectures/starter$ ./if_else_if
enter a character: c
C
input not accepted, please enter a, b, c, or d
cyofanni@LAPTOP-I0S1RKRC:~/Desktop/high-school-cs-class/c_lectures/starter$ ./if_else_if
enter a character: d
D
cyofanni@LAPTOP-I0S1RKRC:~/Desktop/high-school-cs-class/c_lectures/starter$
```

Cos'è successo?

```
if (choice == 'a') {  
    putchar('A');  
    putchar('\n');  
}  
if (choice == 'b') {  
    putchar('B');  
    putchar('\n');  
}  
if (choice == 'c') {  
    putchar('C');  
    putchar('\n');  
}  
if (choice == 'd') {  
    putchar('D');  
    putchar('\n');  
}  
else {  
    puts("input not accepted, please enter a, b, c, or d");  
}
```

**L'utente inserisce 'a'. Il flusso di controllo del programma arriva alla valutazione della condizione puntata dalla freccia.
NB: dobbiamo ancora entrare nel blocco dell'if**

Cos'è successo?

```
if (choice == 'a') {  
    putchar('A');  
    putchar('\n');  
}  
if (choice == 'b') {  
    putchar('B');  
    putchar('\n');  
}  
if (choice == 'c') {  
    putchar('C');  
    putchar('\n');  
}  
if (choice == 'd') {  
    putchar('D');  
    putchar('\n');  
}  
else {  
    puts("input not accepted, please enter a, b, c, or d");  
}
```

**La condizione del primo if viene valutata true.
Il flusso di controllo del programma passa alle
istruzioni del blocco del primo if, che vengono
eseguite tutte**

Cos'è successo?

```
if (choice == 'a') {  
    putchar('A');  
    putchar('\n');  
}  
if (choice == 'b') {  
    putchar('B');  
    putchar('\n');  
}  
if (choice == 'c') {  
    putchar('C');  
    putchar('\n');  
}  
if (choice == 'd') {  
    putchar('D');  
    putchar('\n');  
}  
else {  
    puts("input not accepted, please enter a, b, c, or d");  
}
```

Il flusso di controllo del programma arriva alla valutazione della condizione puntata dalla freccia, che è falsa.
NB: dobbiamo ancora entrare nel blocco dell'if

Cos'è successo?

```
if (choice == 'a') {  
    putchar('A');  
    putchar('\n');  
}  
if (choice == 'b') {  
    putchar('B');  
    putchar('\n');  
}  
if (choice == 'c') {  
    putchar('C');  
    putchar('\n');  
}  
if (choice == 'd') {  
    putchar('D');  
    putchar('\n');  
}  
else {  
    puts("input not accepted, please enter a, b, c, or d");  
}
```

Il flusso di controllo del programma arriva alla valutazione della condizione puntata dalla freccia.
NB: dobbiamo ancora entrare nel blocco dell'if

Cos'è successo?

```
if (choice == 'a') {  
    putchar('A');  
    putchar('\n');  
}  
if (choice == 'b') {  
    putchar('B');  
    putchar('\n');  
}  
if (choice == 'c') {  
    putchar('C');  
    putchar('\n');  
}  
if (choice == 'd') {  
    putchar('D');  
    putchar('\n');  
}  
else {  
    puts("input not accepted, please enter a, b, c, or d");  
}
```

La condizione viene valutata false. Il blocco del terzo if non viene eseguito

Cos'è successo?

```
if (choice == 'a') {  
    putchar('A');  
    putchar('\n');  
}  
if (choice == 'b') {  
    putchar('B');  
    putchar('\n');  
}  
if (choice == 'c') {  
    putchar('C');  
    putchar('\n');  
}  
if (choice == 'd') {  
    putchar('D');  
    putchar('\n');  
}  
else {  
    puts("input not accepted, please enter a, b, c, or d");  
}
```

Il flusso di controllo arriva alla valutazione della condizione del quarto if. La condizione viene valutata false. Il blocco del quarto if non viene eseguito

Cos'è successo?

```
if (choice == 'a') {  
    putchar('A');  
    putchar('\n');  
}  
if (choice == 'b') {  
    putchar('B');  
    putchar('\n');  
}  
if (choice == 'c') {  
    putchar('C');  
    putchar('\n');  
}  
if (choice == 'd') {  
    putchar('D');  
    putchar('\n');  
}  
else {  
    puts("input not accepted, please enter a, b, c, or d");  
}
```

Il blocco dell'else viene eseguito solo se la condizione dell'if precedente era falsa. Effettivamente era falsa, perché 'a' != 'd', quindi il blocco dell'else viene eseguito!!!

Una versione logicamente corretta

```
int main() {  
    char choice;  
    printf("enter a character: ");  
    scanf("%c", &choice);  
    if (choice == 'a') {  
        putchar('A');  
        putchar('\n');  
    }  
    else {  
        if (choice == 'b') {  
            putchar('B');  
            putchar('\n');  
        }  
        else {  
            if (choice == 'c') {  
                putchar('C');  
                putchar('\n');  
            }  
            else {  
                if (choice == 'd') {  
                    putchar('D');  
                    putchar('\n');  
                }  
                else {  
                    puts("input not accepted: please enter a, b, c or d");  
                }  
            }  
        }  
    }  
}
```

Il blocco else puntato dalla freccia blu viene eseguito solo se la condizione dell'if puntato dalla freccia verde è falsa

Una versione logicamente corretta

```
int main() {  
    char choice;  
    printf("enter a character: ");  
    scanf("%c", &choice);  
    if (choice == 'a') {  
        putchar('A');  
        putchar('\n');  
    }  
    else {  
        if (choice == 'b') {  
            putchar('B');  
            putchar('\n');  
        }  
        else {  
            if (choice == 'c') {  
                putchar('C');  
                putchar('\n');  
            }  
            else {  
                if (choice == 'd') {  
                    putchar('D');  
                    putchar('\n');  
                }  
                else {  
                    puts("input not accepted: please enter a, b, c or d");  
                }  
            }  
        }  
    }  
}
```

Il blocco else puntato dalla freccia blu viene eseguito solo se la condizione dell'if puntato dalla freccia verde è falsa

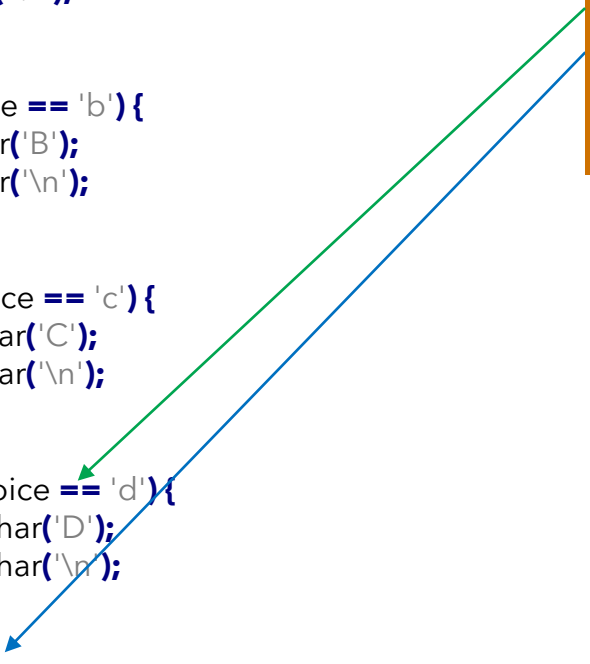
Una versione logicamente corretta

```
int main() {
    char choice;
    printf("enter a character: ");
    scanf("%c", &choice);
    if (choice == 'a') {
        putchar('A');
        putchar('\n');
    }
    else {
        if (choice == 'b') {
            putchar('B');
            putchar('\n');
        }
        else {
            if (choice == 'c') {
                putchar('C');
                putchar('\n');
            }
            else {
                if (choice == 'd') {
                    putchar('D');
                    putchar('\n');
                }
                else {
                    puts("input not accepted: please enter a, b, c or d");
                }
            }
        }
    }
}
```

Il blocco else puntato dalla freccia blu viene eseguito solo se la condizione dell'if puntato dalla freccia verde è falsa

Una versione logicamente corretta

```
int main() {  
    char choice;  
    printf("enter a character: ");  
    scanf("%c", &choice);  
    if (choice == 'a') {  
        putchar('A');  
        putchar('\n');  
    }  
    else {  
        if (choice == 'b') {  
            putchar('B');  
            putchar('\n');  
        }  
        else {  
            if (choice == 'c') {  
                putchar('C');  
                putchar('\n');  
            }  
            else {  
                if (choice == 'd') {  
                    putchar('D');  
                    putchar('\n');  
                }  
                else {  
                    puts("input not accepted: please enter a, b, c or d");  
                }  
            }  
        }  
    }  
}
```



Il blocco else puntato dalla freccia blu viene eseguito solo se la condizione dell'if puntato dalla freccia verde è falsa

Sintassi alternativa per evitare l'annidamento

```
int main() {  
    char choice;  
    printf("enter a character: ");  
    scanf("%c", &choice);  
    if (choice == 'a') {  
        putchar('A');  
        putchar('\n');  
    }  
    else if (choice == 'b') {  
        putchar('B');  
        putchar('\n');  
    }  
    else if (choice == 'c') {  
        putchar('C');  
        putchar('\n');  
    }  
    else if (choice == 'd') {  
        putchar('D');  
        putchar('\n');  
    }  
    else {  
        puts("input not accepted: please enter a, b, c or d");  
    }  
}
```


Una nuova struttura di controllo

- Immaginate un programma che deve stampare una riga di 10 asterischi per 15 volte, con la funzione `printf`
- Con le nostre conoscenze attuali, dovremmo mettere in **sequenza** 15 istruzioni che chiamano la funzione `printf`
- È noiosissimo scrivere programmi in questo modo. Immaginate di dover mettere in sequenza non 15, ma 15 000 istruzioni così...

Una nuova struttura di controllo

```
#include <stdio.h>
```

```
int main() {  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
    printf("*****");  
}
```

**Programmare così è una sofferenza...
Sicuramente per realizzare
programmi poco più che banali ci
serve qualcos'altro**

**Sarebbe utile avere una struttura di
controllo che ci permette di dire al
computer: fai una cosa un certo
numero di volte**

Una nuova struttura di controllo

- Immaginate un programma che deve stampare la tabellina di un numero letto da standard input... per ora potremmo scriverlo così:

```
int main() {  
    int number;  
    printf("enter an integer number: ");  
    scanf("%d", &number);  
  
    puts("the times table of %d is:\n", number);  
    printf("%d\t", number * 1);  
    printf("%d\t", number * 2);  
    printf("%d\t", number * 3);  
    printf("%d\t", number * 4);  
    printf("%d\t", number * 5);  
    printf("%d\t", number * 6);  
    printf("%d\t", number * 7);  
    printf("%d\t", number * 8);  
    printf("%d\t", number * 9);  
    printf("%d\n", number * 10);  
}
```

Una nuova struttura di controllo

- Una versione alternativa potrebbe essere la seguente...

```
int number;
printf("enter an integer number: ");
scanf("%d", &number);
int mult = 1;
printf("the times table of %d is:\n", number);
printf("%d\t", number * mult);
mult++;
printf("%d\t", number * mult);
mult++;
printf("%d\t", number * mult);
mult++;
printf("%d\t", number * mult);
mult++;
printf("%d\t", number * mult);
mult++;
printf("%d\t", number * mult);
mult++;
printf("%d\t", number * mult);
mult++;
printf("%d\t", number * mult);
mult++;
printf("%d\n", number * mult);
```

l'istruzione `mult++` equivale all'istruzione: `mult = mult + 1`
`++` è chiamato operatore di incremento

Iterazione

- Abbiamo bisogno di chiedere al computer di eseguire un blocco di istruzioni un certo numero di volte
- Ci serve una struttura di controllo chiamata **iterazione**
- In C, questa struttura di controllo può essere realizzata tramite la keyword `while`
- La sintassi è analoga a quella dell'costrutto `if`

```
if (boolean condition) {  
}
```

```
while (boolean condition) {  
}
```

Iterazione

```
while (boolean condition) {  
  
}
```

Possiamo tradurlo così in italiano:

**fintantoché la condizione
«boolean condition» è vera, esegui le
istruzioni contenute nel blocco**

Iterazione: cosa fanno queste righe di codice?

```
while (2 == 2) {}
```

```
while (4 < 5);
```

```
while (1) {  
}
```

```
while (0) {  
}
```

Le tabelline

```
/*  
the program prints the times table of  
the integer number read from standard input.  
*/
```

```
int main() {  
    int number, multiplier;  
    multiplier = 1;  
    printf("%s", "enter an integer, the program will show its times table: ");  
    scanf("%d", &number);  
    while (multiplier <= 10) {  
        printf("%d\t", number * multiplier);  
        multiplier++;  
    }  
    printf("\n");  
}
```

Quanto vale multiplier una volta usciti dal ciclo?

11

Le tabelline... codice meno leggibile

```
/*  
the program prints the times table of  
the integer number read from standard input.  
*/
```

```
int main() {  
    int number, multiplier;  
    multiplier = 0;  
    printf("%s", "enter an integer, the program will show its times table: ");  
    scanf("%d", &number);  
    while (multiplier < 10) {  
        printf("%d\t", number * (multiplier + 1));  
        multiplier++;  
    }  
    printf("\n");  
}
```

Quanto vale multiplier una volta usciti dal ciclo?

10

Iterazione: eseguire mentalmente i cicli

```
int k = 0;
```

```
while (k < 10){  
    k += 1;  
}
```

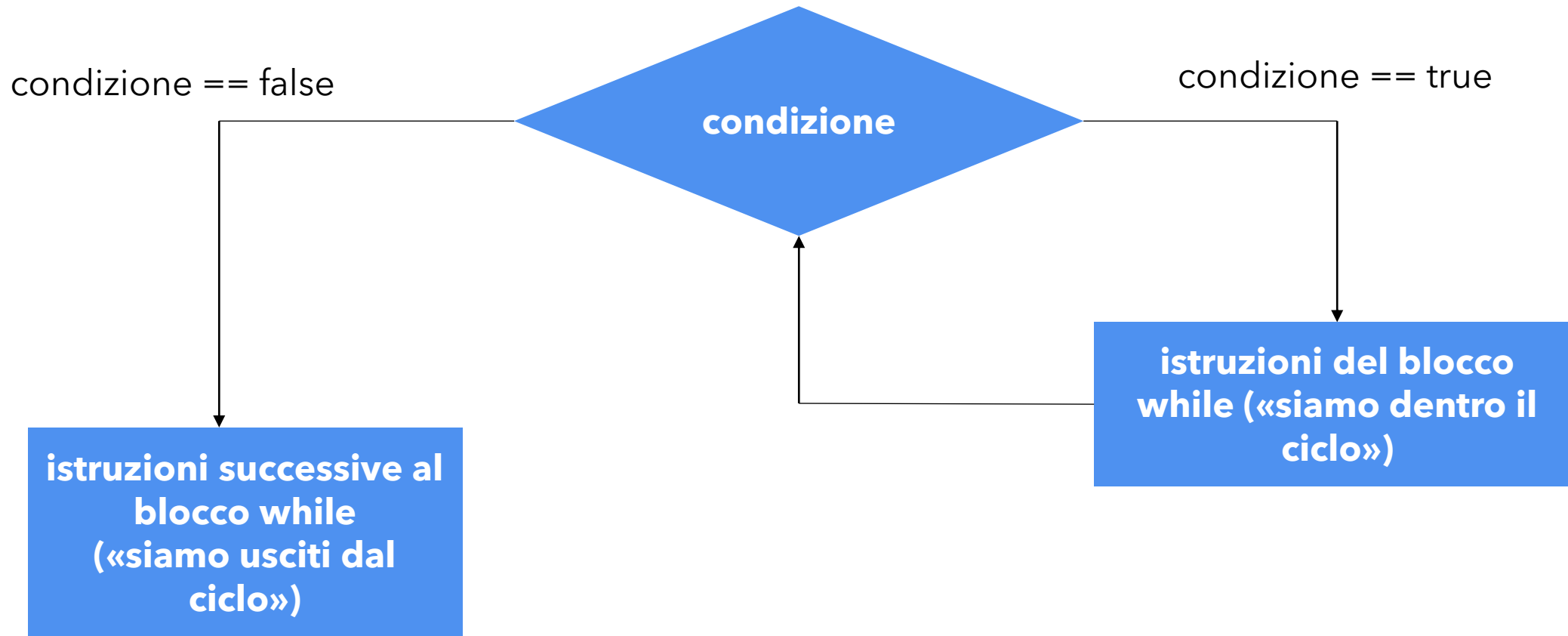
```
int k = 10;
```

```
while (k > 0){  
    k -= 1;  
}
```

Iterazione: eseguire mentalmente i cicli

```
int k = 1;  
int s = 0;  
  
while (k <= 100) {  
    s += k;  
    k += 1;  
}
```

L'iterazione con i diagrammi di flusso



Iterazione controllata da un valore sentinella

/*

the program computes the sum of a sequence of numbers
read from stdin.

When the user enters the flag -1, the program prints the sum.

*/

Riflettete prima di lanciaarvi nella scrittura del codice.

Quante variabili serviranno?

Quali strutture di controllo serviranno?

Iterazione controllata da un valore sentinella

/*

the program computes the product of a sequence of numbers read from stdin.

When the user enters the flag -1, the program prints the sum.

*/

Riflettete prima di lanciarsi nella scrittura del codice.

Quante variabili serviranno?

Quali strutture di controllo serviranno?

“Srotoliamo” il ciclo

Stato del programma prima di testare la condizione del ciclo per la prima volta

Variabile	Valore
total_product	1
i	-99999

→ Il controllo passa al test di permanenza del ciclo

→ La condizione di permanenza del ciclo è **falsa** ((i == -99999) != -99999 ha valore **false**) →

→ Si «esce» dal ciclo, ossia il controllo passa all'istruzione successiva al ciclo

“Srotoliamo” il ciclo

Stato del programma prima di testare la condizione del ciclo per la prima volta

Variabile	Valore
total_product	1
i	8



Il controllo passa al test di permanenza del ciclo



La condizione di permanenza del ciclo è **vera** ((i == 8) != -99999 ha valore **true**)



Verrà eseguita un'altra iterazione

“Srotoliamo” il ciclo

Stato del programma al termine
dell'iterazione 1

Variabile	Valore
total_product	total_product*i: 1*8 = 8
i	5

Da dove vengono questi
valori?

Il controllo passa al test di permanenza
del ciclo

La condizione di permanenza del ciclo è
vera ((i == 5) != -99999 ha valore **true**)

Verrà eseguita un'altra iterazione

“Srotoliamo” il ciclo

Stato del programma al termine
dell'iterazione 2

Variabile	Valore
total_product	total_product*i: 8*5 = 40
i	3

Da dove vengono questi
valori?

Il controllo passa al test di permanenza
del ciclo

La condizione di permanenza del ciclo è
vera ((i == 3) != -99999 ha valore **true**)

Verrà eseguita un'altra iterazione

“Srotoliamo” il ciclo

Stato del programma al termine
dell'iterazione 3

Variabile	Valore
total_product	total_product*i: 40*3 =120
i	2

Da dove vengono questi
valori?

Il controllo passa al test di permanenza
del ciclo

La condizione di permanenza del ciclo è
vera ((i == 2) != -99999 ha valore **true**)

Verrà eseguita un'altra iterazione

“Srotoliamo” il ciclo

Stato del programma al termine
dell'iterazione 4

Variabile	Valore
total_product	total_product*i: 120*2 = 240
i	-99999

Da dove vengono questi
valori?

Il controllo passa al test di permanenza
del ciclo

La condizione di permanenza del ciclo è
falsa ((i == -99999) != -99999 ha valore
false)

Si «esce» dal ciclo: il controllo passa
all'istruzione successiva al ciclo

Una rappresentazione alternativa

	total_product	i (standard_input)
termine iterazione 0 (prima di valutare la condizione del ciclo la prima volta)	1	-99999

→ La condizione di permanenza viene valutata **false**

Una rappresentazione alternativa

	total_product	i (standard_input)	
termine iterazione 0 (prima di valutare la condizione del ciclo la prima volta)	1	8	→ La condizione di permanenza viene valutata true
termine iterazione 1	$1 * 8 = 8$	5	→ La condizione di permanenza viene valutata true
termine iterazione 2	$8 * 5 = 40$	3	→ La condizione di permanenza viene valutata true
termine iterazione 3	$40 * 3 = 120$	2	→ La condizione di permanenza viene valutata true
termine iterazione 4	$120 * 2 = 240$	-99999	→ La condizione di permanenza viene valutata false

Iterazione

```
int previous = -1;  
int input;  
scanf("%d", &input);  
  
while (input != -1 && input != previous) {  
    previous = input;  
    scanf("%d", &input);  
}
```

Srotolare!

fine iterazione 0

previous

**input
(stdin)**

-1

-1

→ La condizione di
permanenza viene
valutata **false**

Srotolare!

fine iterazione 0	previous	input (stdin)	→ La condizione di permanenza viene valutata true			
	-1	5				
fine iterazione 1		previous	input (stdin)	→ La condizione di permanenza viene valutata true		
			7			
fine iterazione 2			previous	input (stdin)	→ La condizione di permanenza viene valutata true	
				8		
fine iterazione 3			previous	input (stdin)	→ La condizione di permanenza viene valutata true	
				4		
fine iterazione 4				previous	input (stdin)	→ La condizione di permanenza viene valutata false
					-1	

Srotolare!

fine iterazione 0	previous	input (stdin)	→ La condizione di permanenza viene valutata true				
	-1	5					
fine iterazione 1		previous	input (stdin)	→ La condizione di permanenza viene valutata true			
			7				
fine iterazione 2			previous	input (stdin)	→ La condizione di permanenza viene valutata true		
				8			
fine iterazione 3				previous	input (stdin)	→ La condizione di permanenza viene valutata true	
					4		
fine iterazione 4					previous	input (stdin)	→ La condizione di permanenza viene valutata false
						4	

Iterazione

1. Scrivere un programma che accetti in input una successione di interi finché è verificata la condizione seguente: l'intero in input è diverso da 99999 e l'ultimo intero inserito è maggiore o uguale al precedente;
2. Scrivere un programma che accetti in input una successione di interi finché è verificata la condizione seguente: l'intero in input è diverso da 99999 e l'ultimo intero inserito è minore del precedente;
3. Scrivere un programma che accetti in input una successione di interi finché è verificata la condizione seguente: l'intero in input è diverso da 99999 e l'ultimo intero inserito è uguale al doppio del precedente. La condizione di permanenza del ciclo deve essere vera almeno la prima volta che viene valutata;
4. Scrivere un programma che accetti in input una successione di interi finché è verificata la condizione seguente: l'intero in input è diverso da 99999 e l'ultimo intero inserito è uguale al quadrato del precedente. La condizione di permanenza del ciclo deve essere vera almeno la prima volta che viene valutata;
5. Scrivere un programma che memorizzi il valore massimo di una successione di interi letta da standard input finché è verificata la condizione seguente: l'intero in input è diverso da 99999;

Teorema di Boehm-Jacopini (1966)

Qualsiasi algoritmo può essere realizzato tramite le seguenti 3 strutture di controllo:

- **sequenza;**
- **selezione** (detta anche **ramificazione**);
- **iterazione**

Teorema di Boehm-Jacopini (1966)

Qualsiasi algoritmo può essere realizzato tramite un diagramma di flusso strutturato come:

- **sequenza di sotto-diagrammi di flusso;**
- **selezione tra sotto-diagrammi di flusso;**
- **iterazione di un sotto-diagramma di flusso**