

Esercitazione 03

2025/10/01 – Strutture di Controllo (LOOP)

L. Alessandrini

Credits: A. Montenegro

Esercizi parte 1 di 2

While – Do While

Notazione per gli incrementi

Differenza `i++` e `++i`

`i++` e `++i` sono due modi diversi di incrementare il valore di una variabile.

Notazione per gli

Differenza i++ e ++i

i++ e ++i sono due modi diversi di incrementare il valore di una variabile.

```
Valore iniziale di i: 0
Valore di a: 0
Valore di i dopo i++: 1
Valore di i dopo a = ++i: 2
Valore di a dopo a = ++i: 2
Valore di i dopo a = i++: 2
Valore di a dopo a = i++: 1
```

```
#include <stdio.h>
int main(){
    // differenza tra i++ e ++i
    int i=0, a=0;

    printf("Valore iniziale di i: %d\n", i);
    printf("Valore di a: %d\n", a);

    i++;

    printf("Valore di i dopo i++: %d\n", i);

    a = ++i; // equivalente a i = i + 1; a = i;

    printf("Valore di i dopo a = ++i: %d\n", i);
    printf("Valore di a dopo a = ++i: %d\n", a);

    i = 1;

    a = i++; // equivalente a a = i; i = i + 1;
    printf("Valore di i dopo a = i++: %d\n", i);
    printf("Valore di a dopo a = i++: %d\n", a);

    return 0;
}
```

Notazione per gli

Differenza i++ e ++i

i++ e ++i sono due modi diversi di incrementare il valore

Equivale a $i = i + 1$

```
Valore iniziale di i: 0
Valore di a: 0
Valore di i dopo i++: 1
Valore di i dopo a = ++i: 2
Valore di a dopo a = ++i: 2
Valore di i dopo a = i++: 2
Valore di a dopo a = i++: 1
```

```
#include <stdio.h>
int main(){
    // differenza tra i++ e ++i
    int i=0, a=0;

    printf("Valore iniziale di i: %d\n", i);
    printf("Valore di a: %d\n", a);

    i++;

    printf("Valore di i dopo i++: %d\n", i);

    a = ++i; // equivalente a i = i + 1; a = i;

    printf("Valore di i dopo a = ++i: %d\n", i);
    printf("Valore di a dopo a = ++i: %d\n", a);

    i = 1;

    a = i++; // equivalente a a = i; i = i + 1;
    printf("Valore di i dopo a = i++: %d\n", i);
    printf("Valore di a dopo a = i++: %d\n", a);

    return 0;
}
```

Notazione per gli

Differenza i++ e ++i

i++ e ++i sono due modi diversi di incrementare il valore

Equivale a $i = i + 1$

Valore iniziale di i: 0

Valore di a: 0

Valore di i dopo i++: 1

Valore di i dopo a = ++i: 2

Valore di a dopo a = ++i: 2

Valore di i dopo a = i++: 2

Valore di a dopo a = i++: 1

```
#include <stdio.h>
int main(){
    // differenza tra i++ e ++i
    int i=0, a=0;

    printf("Valore iniziale di i: %d\n", i);
    printf("Valore di a: %d\n", a);

    i++;

    printf("Valore di i dopo i++: %d\n", i);

    a = ++i; // equivalente a i = i + 1; a = i;

    printf("Valore di i dopo a = ++i: %d\n", i);
    printf("Valore di a dopo a = ++i: %d\n", a);

    i = 1;

    a = i++; // equivalente a a = i; i = i + 1;
    printf("Valore di i dopo a = i++: %d\n", i);
    printf("Valore di a dopo a = i++: %d\n", a);

    return 0;
}
```

Notazione per gli

Differenza i++ e ++i

i++ Prima incrementa i
div (ora varrà 2), poi lo
val assegna ad a

```
Valore iniziale di i: 0
Valore di a: 0
Valore di i dopo i++: 1
Valore di i dopo a = ++i: 2
Valore di a dopo a = ++i: 2
Valore di i dopo a = i++: 2
Valore di a dopo a = i++: 1
```

```
#include <stdio.h>
int main(){
    // differenza tra i++ e ++i
    int i=0, a=0;

    printf("Valore iniziale di i: %d\n", i);
    printf("Valore di a: %d\n", a);

    i++;

    printf("Valore di i dopo i++: %d\n", i);

    a = ++i; // equivalente a i = i + 1; a = i;

    printf("Valore di i dopo a = ++i: %d\n", i);
    printf("Valore di a dopo a = ++i: %d\n", a);

    i = 1;

    a = i++; // equivalente a a = i; i = i + 1;
    printf("Valore di i dopo a = i++: %d\n", i);
    printf("Valore di a dopo a = i++: %d\n", a);

    return 0;
}
```

Notazione per gli

Differenza i++ e ++i

i++
div
val

Prima incrementa i
(ora varrà 2), poi lo
assegna ad a

Valore iniziale di i: 0

Valore di a: 0

Valore di i dopo i++: 1

Valore di i dopo a = ++i: 2

Valore di a dopo a = ++i: 2

Valore di i dopo a = i++: 2

Valore di a dopo a = i++: 1

```
#include <stdio.h>
int main(){
    // differenza tra i++ e ++i
    int i=0, a=0;

    printf("Valore iniziale di i: %d\n", i);
    printf("Valore di a: %d\n", a);

    i++;

    printf("Valore di i dopo i++: %d\n", i);

    a = ++i; // equivalente a i = i + 1; a = i;

    printf("Valore di i dopo a = ++i: %d\n", i);
    printf("Valore di a dopo a = ++i: %d\n", a);

    i = 1;

    a = i++; // equivalente a a = i; i = i + 1;
    printf("Valore di i dopo a = i++: %d\n", i);
    printf("Valore di a dopo a = i++: %d\n", a);

    return 0;
}
```

Notazione per gli

Differenza i++ e ++i

i++ e ++i sono due modi diversi di incrementare il valore di una variabile.

Prima assegna il valore di i ad a, poi incrementa i

```
Valore iniziale di i: 0
Valore di a: 0
Valore di i dopo i++: 1
Valore di i dopo a = ++i: 2
Valore di a dopo a = ++i: 2
Valore di i dopo a = i++: 2
Valore di a dopo a = i++: 1
```

```
#include <stdio.h>
int main(){
    // differenza tra i++ e ++i
    int i=0, a=0;

    printf("Valore iniziale di i: %d\n", i);
    printf("Valore di a: %d\n", a);

    i++;

    printf("Valore di i dopo i++: %d\n", i);

    a = ++i; // equivalente a i = i + 1; a = i;

    printf("Valore di i dopo a = ++i: %d\n", i);
    printf("Valore di a dopo a = ++i: %d\n", a);

    i = 1;

    a = i++; // equivalente a a = i; i = i + 1;
    printf("Valore di i dopo a = i++: %d\n", i);
    printf("Valore di a dopo a = i++: %d\n", a);

    return 0;
}
```

Notazione per gli

Differenza i++ e ++i

i++ e ++i sono due modi diversi di incrementare il valore di una variabile.

Prima assegna il valore di i ad a, poi incrementa i

```
Valore iniziale di i: 0
Valore di a: 0
Valore di i dopo i++: 1
Valore di i dopo a = ++i: 2
Valore di a dopo a = ++i: 2
Valore di i dopo a = i++: 2
Valore di a dopo a = i++: 1
```

```
#include <stdio.h>
int main(){
    // differenza tra i++ e ++i
    int i=0, a=0;

    printf("Valore iniziale di i: %d\n", i);
    printf("Valore di a: %d\n", a);

    i++;

    printf("Valore di i dopo i++: %d\n", i);

    a = ++i; // equivalente a i = i + 1; a = i;

    printf("Valore di i dopo a = ++i: %d\n", i);
    printf("Valore di a dopo a = ++i: %d\n", a);

    i = 1;

    a = i++; // equivalente a a = i; i = i + 1;
    printf("Valore di i dopo a = i++: %d\n", i);
    printf("Valore di a dopo a = i++: %d\n", a);

    return 0;
}
```

Esercizio 01

Inserimento Sequenziale di Caratteri con Criterio di Stop

Testo

Si scriva un programma che permetta di inserire all'utente un carattere alla volta. Ogni volta che l'utente inserisce un carattere il programma deve stampare:

1. il numero di caratteri inseriti finora;
2. l'ultimo carattere inserito.

Il programma termina quando l'utente inserisce il carattere '*'.

Esercizio 02

Stampa delle Tabelline

Testo

Si scriva un programma in grado di stampare le tabelline (da 1 a 10) usando solo cicli while.

Esercizio 03

Calcolo delle Cifre di un Numero

Testo

Si scriva un programma che permetta all'utente di inserire un numero intero N in input (positivo o negativo). Il programma calcola il numero di cifre di N e lo stampa a schermo.

Esempio

1234 → 4

Esercizio 04

Inversione delle Cifre di un Numero

Testo

Si scriva un programma che permetta all'utente di inserire un numero intero positivo N in input. Il programma deve stampare a schermo il numero inserito dall'utente con tutte le cifre invertite.

Esempio

1234 → 4321

Esercizio 06

Calcolatrice ... parte 1!

Testo

Si scriva un programma che replichi il funzionamento di una calcolatrice. Implementare solamente le 4 operazioni: '+', '-', '*', '/'.

Switch

Statement Switch per l'esecuzione di blocchi di codice

Switch Statement

Parte 1 di 3

- In C il costrutto **switch** è utilizzato per eseguire alcuni blocchi di codice in base al valore di un'espressione
- Utile quando ci sono molte condizioni basate sul valore di una singola variabile
- Sostituibile con una serie di **if-else**

Switch Statement

Parte 2 di 3

- Keywords:
 - **switch**
 - **case**
 - **default**
- Espressioni:
 - **int_expr**: espressione a valori int oppure char (non booleani)
 - **constant_expr**: numero oppure carattere (non variabili)
- Il caso **default** è opzionale

```
switch(int_expr){  
    case constant-expr1:  
        statement-1;  
    case constant-expr2:  
        statement-2;  
    ...  
    case constant-exprN:  
        statement-N;  
    [default:  
        statement;]  
}
```

Switch Statement

Parte 3 di 3

Gli **statement** di ogni caso non hanno bisogno di essere delimitati da `{}` anche nel caso in cui contengano più istruzioni

Gli **statement** di ogni **case** sono limitati dal **case** successivo (o dal **default**, se presente)

```
switch(int_expr){  
    case constant-expr1:  
        statement-1;  
    case constant-expr2:  
        statement-2;  
    ...  
    case constant-exprN:  
        statement-N;  
    [default:  
        statement;]  
}
```

Switch Statement

Esecuzione

1. Viene valutata **int_expr**
2. Si controlla se **int_expr** è uguale a **constant-expr-1**
 1. Se sono uguali viene eseguito **statement-1**, poi in cascata vengono eseguiti tutti gli altri **statement** relativi agli altri casi (senza verificare l'uguaglianza delle espressioni)
 2. Se NON sono uguali, si controlla se **int_expr** è uguale a **constant-expr-2** [...]
3. Si esegue lo **statement** nel caso **default**

Per evitare l'esecuzione in cascata, si può usare la keyword **break**!

```
switch(int_expr){  
    case constant-expr1:  
        statement-1;  
    case constant-expr2:  
        statement-2;  
    ...  
    case constant-exprN:  
        statement-N;  
    [default:  
        statement;]  
}
```

Switch Statement

Esecuzione

1. Viene valutata **int_expr**
2. Si controlla se **int_expr** è uguale a **constant-expr-1**

1. Se

poi

sta

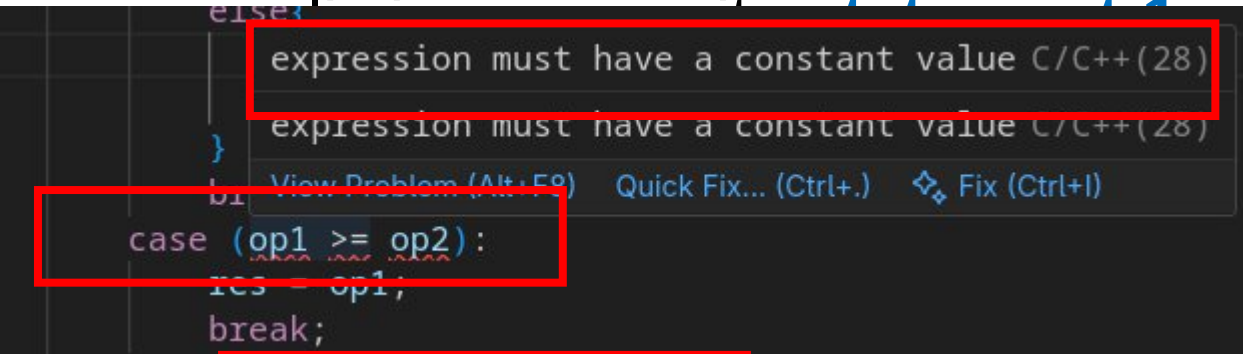
veri

2. Se

è uguale a **constant-expr-2** [...]

3. Si esegue lo **statement** nel caso

Devono essere costanti!



```
switch(int_expr){
    case constant-expr1:
        statement-1;
    case constant-expr2:
        statement-2;
    ...
    case constant-exprN:
        statement-N;
    [default:
        statement;]
}
```

Per evitare l'esecuzione in cascata, si può usare la keyword **break**!

Switch Statement

Switch: to break or not to break?

```
int main(int argc, char* argv){
    char c;
    int nA = 0, nE = 0, nCons = 0;

    scanf("%c", &c);
    switch(c){
        case 'a':
            nA++;
            break;
        case 'e':
            nE++;
            break;
        default:
            nCons++;
            break;
    }

    printf("%d %d %d\n", nA, nE, nCons);
    return 0;
}
```

```
int main(int argc, char* argv){
    char c;
    int nA = 0, nE = 0, nCons = 0;

    scanf("%c", &c);
    switch(c){
        case 'a':
            nA++;
            // break;
        case 'e':
            nE++;
            // break;
        default:
            nCons++;
            // break;
    }

    printf("%d %d %d\n", nA, nE, nCons);
    return 0;
}
```

Switch Statement

Switch: to break or not to break?

```
int main(int argc, char* argv){
    char c;
    int nA = 0, nE = 0, nCons = 0;

    scanf("%c", &c);
    switch(c){
        case 'a':
            nA++;
            break;
        case 'e':
            nE++;
            break;
        default:
            nCons++;
            break;
    }

    printf("%d %d %d\n", nA, nE, nCons);
    return 0;
}
```

Che
risultato ci
aspettiamo
?

```
int main(int argc, char* argv){
    char c;
    int nA = 0, nE = 0, nCons = 0;

    scanf("%c", &c);
    switch(c){
        case 'a':
            nA++;
            // break;
        case 'e':
            nE++;
            // break;
        default:
            nCons++;
            // break;
    }

    printf("%d %d %d\n", nA, nE, nCons);
    return 0;
}
```

Switch Statement - Calcolatrice

Esempio moltiplicazione senza break

Switch Statement - Cal

Esempio moltiplicazione senza break

```
#include <stdio.h>
int main(int argc, char* argv[]){
    // Variabili
    float op1, op2, res;
    char operazione;
    // Acquisizione da tastiera del primo operando
    printf("Inserire il PRIMO operando: ");
    scanf("%f", &op1);
    scanf("%c"); // consumo \n in quanto dovrò leggere un char
    // Acquisizione da tastiera dell'operazione
    printf("Inserire l'operazione da eseguire: ");
    scanf("%c", &operazione);
    // Acquisizione da tastiera del secondo operando
    printf("Inserire il SECONDO operando: ");
    scanf("%f", &op2);

    // Svolgo l'operazione
    switch(operazione){
        case '+':
            res = op1 + op2;
            printf("addizione\n");
        case '-':
            res = op1 - op2;
            printf("sottrazione\n");
        case '*':
            res = op1 * op2;
            printf("moltiplicazione\n");
        case '/':
            printf("divisione\n");
            if(op2 != 0) res = op1 / op2;
            else{
                printf("Errore!\n");
                return 1;
            }
        default:
            printf("Operazione non consentita!\n");
            return 1;
    }

    printf("%.2f %c %.2f = %.2f.\n", op1, operazione, op2, res);

    return 0;
}
```

Switch Statement - Cal

Esempio moltiplicazione senza break

```
Inserire il PRIMO operando: 2
Inserire l'operazione da eseguire: *
Inserire il SECONDO operando: 3
moltiplicazione
divisione
Operazione non consentita!
```

```
#include <stdio.h>
int main(int argc, char* argv[]){
    // Variabili
    float op1, op2, res;
    char operazione;
    // Acquisizione da tastiera del primo operando
    printf("Inserire il PRIMO operando: ");
    scanf("%f", &op1);
    scanf("%c"); // consumo \n in quanto dovrò leggere un char
    // Acquisizione da tastiera dell'operazione
    printf("Inserire l'operazione da eseguire: ");
    scanf("%c", &operazione);
    // Acquisizione da tastiera del secondo operando
    printf("Inserire il SECONDO operando: ");
    scanf("%f", &op2);

    // Svolgo l'operazione
    switch(operazione){
        case '+':
            res = op1 + op2;
            printf("addizione\n");
        case '-':
            res = op1 - op2;
            printf("sottrazione\n");
        case '*':
            res = op1 * op2;
            printf("moltiplicazione\n");
        case '/':
            printf("divisione\n");
            if(op2 != 0) res = op1 / op2;
            else{
                printf("Errore!\n");
                return 1;
            }
        default:
            printf("Operazione non consentita!\n");
            return 1;
    }

    printf("%.2f %c %.2f = %.2f.\n", op1, operazione, op2, res);

    return 0;
}
```

Switch Statement -

Esempio moltiplicazione senza l

```
Inserire il PRIMO operando: 2
Inserire l'operazione da eseguire: *
Inserire il SECONDO operando: 3
moltiplicazione
divisione
Operazione non consentita!
```

Sono stati eseguiti i contenuti di tutti gli statement, a partire dalla moltiplicazione, incluso il default.

Siccome nel default inoltre ritorno un errore con la return, non arrivo ad eseguire la print del risultato

```
#include <stdio.h>

printf("addizione\n");
case '-':
    res = op1 - op2;
    printf("sottrazione\n");
case '*':
    res = op1 * op2;
    printf("moltiplicazione\n");
case '/':
    printf("divisione\n");
    if(op2 != 0) res = op1 / op2;
    else{
        printf("Errore!\n");
        return 1;
    }
default:
    printf("Operazione non consentita!\n");
    return 1;
}

printf("%.2f %c %.2f = %.2f.\n", op1, operazione, op2, res);

return 0;
}
```

Switch Statement - Cal

Esempio moltiplicazione senza break

Nota come i casi debbano essere constant expressions, quindi non potrò scrivere ad esempio " ≥ 10 ", e nemmeno usare i doppi apici, ma solamente i singoli

```
I  
I  
I  
m  
divisione  
Operazione non consentita!
```

```
#include <stdio.h>  
int main(int argc, char* argv[]){  
    // Variabili  
    float op1, op2, res;  
    char operazione;  
    // Acquisizione da tastiera del primo operando  
    printf("Inserire il PRIMO operando: ");  
    scanf("%f", &op1);  
    scanf("%c"); // consumo \n in quanto dovrò leggere un char  
    // Acquisizione da tastiera dell'operazione  
    printf("Inserire l'operazione da eseguire: ");  
    scanf("%c", &operazione);  
    // Acquisizione da tastiera del secondo operando  
    printf("Inserire il SECONDO operando: ");  
    scanf("%f", &op2);
```

```
    // Svolgo l'operazione  
    switch(operazione){  
        case '+':  
            res = op1 + op2;  
            printf("addizione\n");  
        case '-':  
            res = op1 - op2;  
            printf("sottrazione\n");  
        case '*':  
            res = op1 * op2;  
            printf("moltiplicazione\n");  
        case '/':  
            printf("divisione\n");  
            if(op2 != 0) res = op1 / op2;  
            else{  
                printf("Errore!\n");  
                return 1;  
            }  
        default:  
            printf("Operazione non consentita!\n");  
            return 1;  
    }  
  
    printf("%.2f %c %.2f = %.2f.\n", op1, operazione, op2, res);  
  
    return 0;  
}
```

Switch Statement - Cal

Esempio moltiplicazione senza break

Nota come i casi debbano essere constant expressions, quindi non potrò scrivere ad esempio " ≥ 10 ", e nemmeno usare i doppi apici, ma solamente i singoli

```
#include <stdio.h>
int main(int argc, char* argv[]){
    // Variabili
    float op1, op2, res;
    char operazione;
    // Acquisizione da tastiera del primo operando
    printf("Inserire il PRIMO operando: ");
    scanf("%f", &op1);
    scanf("%c"); // consumo \n in quanto dovrò leggere un char
    // Acquisizione da tastiera dell'operazione
    printf("Inserire l'operazione da eseguire: ");
    scanf("%c", &operazione);
    // Acquisizione da tastiera del secondo operando
    printf("Inserire il SECONDO operando: ");
    scanf("%f", &op2);
```

```
    // Svolgo l'operazione
    switch(operazione){
        case '+':
            res = op1 + op2;
            printf("addizione\n");
        case '-':
            res = op1 - op2;
            printf("sottrazione\n");
        case '*':
            res = op1 * op2;
            printf("moltiplicazione\n");
        case '/':
            printf("divisione\n");
            if(op2 != 0) res = op1 / op2;
            else{
                printf("Errore!\n");
                return 1;
            }
        default:
            printf("Operazione non consentita!\n");
            return 1;
    }
```

```
    printf("%.2f %c %.2f = %.2f.\n", op1, operazione, op2, res);
    return 0;
}
```

expression must be an integral constant expression C/C++(157)

View Problem (Alt+F8) Quick Fix... (Ctrl+.) Fix (Ctrl+I)

case "/":

expected an expression C/C++(29)

View Problem (Alt+F8) Quick Fix... (Ctrl+.) Fix (Ctrl+I)

case ≥ 3 :

Esercizi parte 2 di 2

While – Do While

Esercizio 07

Calcolatrice ... parte 2!

Testo

Si scriva un programma che replichi il funzionamento di una calcolatrice. Implementare solamente le 4 operazioni: '+', '-', '*', '/'. Usare il costrutto switch.

Thank You!

Mail: luca1.alessandrini@polimi.it

Website: <https://alessandriniluca.github.io/>