

FONDAMENTI DI PROGRAMMAZIONE

PROGRAMMAZIONE STRUTTURATA IN C

Dott. Federico Concone

federico.concone@unipa.it

Informazioni utili

- Ricevimento:
 - Edificio 6, 3° Piano, Laboratorio Intelligenza Artificiale e Sistemi Distribuiti;
- Orario ricevimento:
 - Da concordare tramite e-mail;
- Testo consigliato:
 - Deitel, Deitel, Il linguaggio C – Fondamenti e tecniche di programmazione 8 Ed., Pearson, Italia, 2016
- Link materiale:
 - goo.gl/mrcWL7

Informazioni utili

- COMUNICAZIONE PER GLI **STUDENTI ISCRITTI AD ANNI SUCCESSIVI AL PRIMO**;
- ISCRIZIONE ALLA PROVA IN **ITINERE DEL 18/04/2018** CON DOCENTE DI RIFERIMENTO **MARCO ORTOLANI**:
 - FONDAMENTI DI PROGRAMMAZIONE IN C;
 - ARCHITETTURE AVANZATE DEI CALCOLATORI;
- PER POTERSI ISCRIVERE **INVIARE UNA E-MAIL** CON I SEGUENTI CAMPI:
 1. *Destinatari:* marco.ortolani@unipa.it e federico.concone@unipa.it;
 2. *Oggetto:* iscrizione alla prova in itinere di giorno 18/04/2018;
 3. *Corpo:* Nome, Cognome, Matricola, Anno di corso;
- LA PROVA DEL 19/04/2018 E' RISERVATA AI FUORI CORSO;

Sommario

- Istruzioni speciali break e continue
- Strutture di controllo
 - Strutture di condizione
 - Strutture di iterazione
- Le variabili di flag
- I cicli annidati
- Esercizi

Strutture di controllo

- Il linguaggio C fornisce 4 tipi di *strutture di selezione* e 3 *strutture di ripetizione*:

- **If** statement;
- **If...else** statement;
- **Nested If...else** statement;
- **Switch** statement;



Strutture di condizione

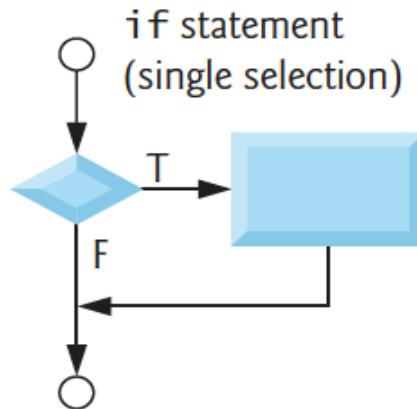
- **While** statement;
- **Do...while** statement;
- **For** statement;



Strutture di iterazione

Istruzioni speciali

- Prima di introdurre il costrutto **switch** è necessario conoscere due istruzioni molto importanti: **break** e **continue**;
- Queste speciali istruzioni interrompono il normale flusso del controllo;
- Il flusso di controllo per il costrutto if è il seguente:



Istruzioni speciali: break

- L'istruzione **break** causa l'uscita dal ciclo più interno che la contiene o da un'istruzione switch;
- Un tipico utilizzo di **break** è il seguente:

```
while(1) {  
    scanf(“%lf”, &x);  
    if (x < 0.0)  
        break;  
    printf(“%f\n”, x + 1);  
}  
  
/* break salta a questo punto */
```

- Quello che sarebbe stato un ciclo infinito termina quando la condizione dell'**if** risulta vera;

Istruzioni speciali: continue

- L'istruzione **continue** interrompe l'esecuzione dell'iterazione corrente del ciclo e causa l'inizio dell'iterazione successiva;
- Un esempio dell'utilizzo di **continue** è il seguente:

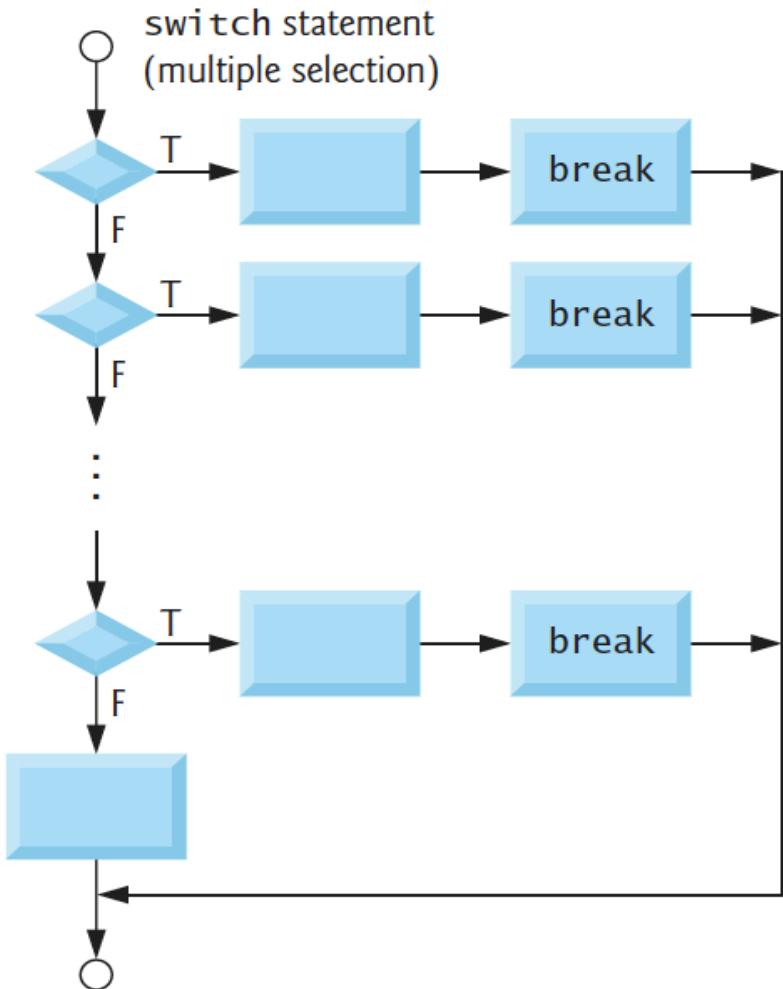
```
while( x<10) {  
    i = x - 4;  
    if ( i < 0.0)  
        continue;  
    printf(“%f\n”, i + 1);  
    x++;  
    /* continue trasferisce il controllo in questo punto */  
}
```

- L'istruzione **continue** può trovarsi solo all'interno dei cicli **for**, **while** e **do-while**;

Strutture di selezione: SWITCH

- Lo statement *switch* generalizza l'istruzione *if...else*;

```
switch(expression) {  
    case(constant-expression):  
        statement_set_1  
        break;  
    case(constant-expression):  
        statement_set_2  
        break;  
    default:  
        statement_set_3  
        break;  
}
```



Strutture di selezione: SWITCH

- Lo statement speciale *break* causa l'uscita dal ciclo più interno che lo contiene, interrompendo il normale flusso di controllo;

```
switch(expression) {  
    case(constant-expression):  
        statement_set_1  
  
        ←  
    case(constant-expression):  
        statement_set_2  
        break;  
    default:  
        statement_set_3  
        break;  
}
```

**COSA SUCCIDE
SE TOLGO IL
BREAK?**

Strutture di selezione: SWITCH

- Lo statement speciale *break* causa l'uscita dal ciclo più interno che lo contiene, interrompendo il normale flusso di controllo;

```
switch(expression) {  
    case(constant-expression):  
        statement_set_1  
  
    case(constant-expression):  
        statement_set_2  
        break;  
    default:  
        statement_set_3  
        break;  
}
```

**VERRANNO ESEGUITE
LE ISTRUZIONI DEI
CASI SOTTOSTANTI FIN
TANTO CHE NON
VIENE INCONTRATO
UN BREAK**

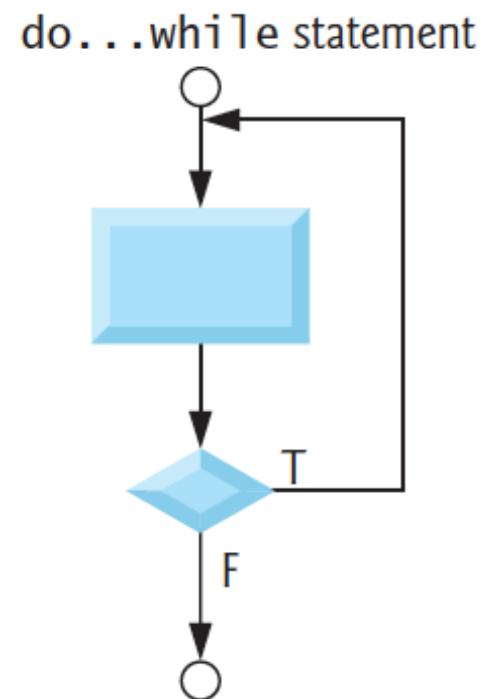
Strutture di selezione: SWITCH

```
1 #include<stdio.h>
2
3 int main(void){
4     char character;
5     printf("Inserire un carattere [a-d]: \n");
6     scanf("%c", &character);
7
8     switch(character){
9         case 'a':
10            printf("Hai inserito il carattere: %c\n", character);
11            break;
12        case 'b':
13            printf("Hai inserito il carattere: %c\n", character);
14
15        case 'c':
16            printf("Hai inserito il carattere: %c\n", character);
17
18        case 'd':
19            printf("Hai inserito il carattere: %c\n", character);
20            break;
21        default:
22            printf("Carattere non valido\n");
23            break;
24    }
25    return 0;
26 }
```

Strutture di iterazione: DO... WHILE

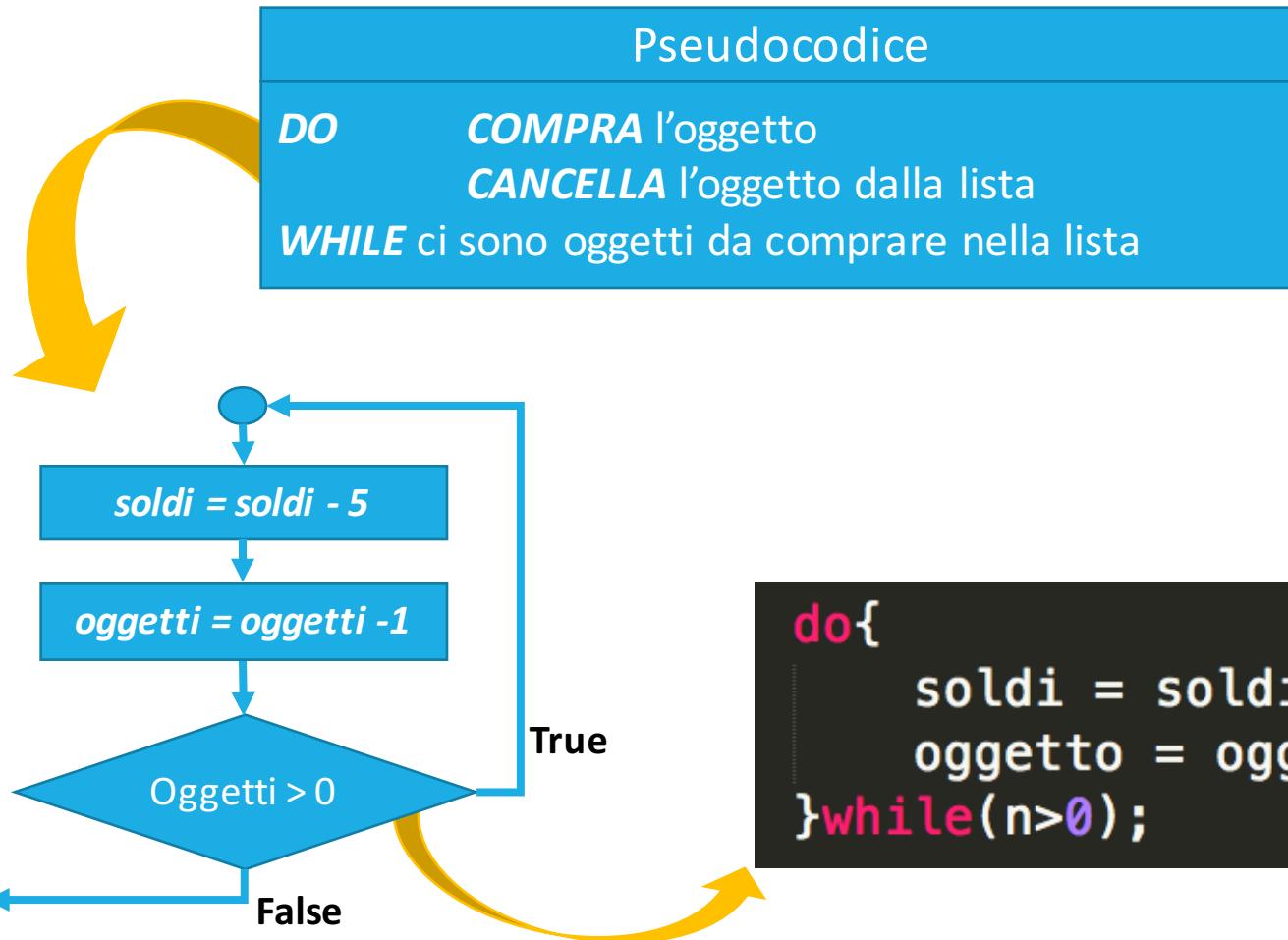
- Lo statement do...while ha un **comportamento simile al while ma esegue l'azione almeno una volta;**

```
do {  
    statement  
} while(expression) ;
```



Strutture di iterazione: DO... WHILE

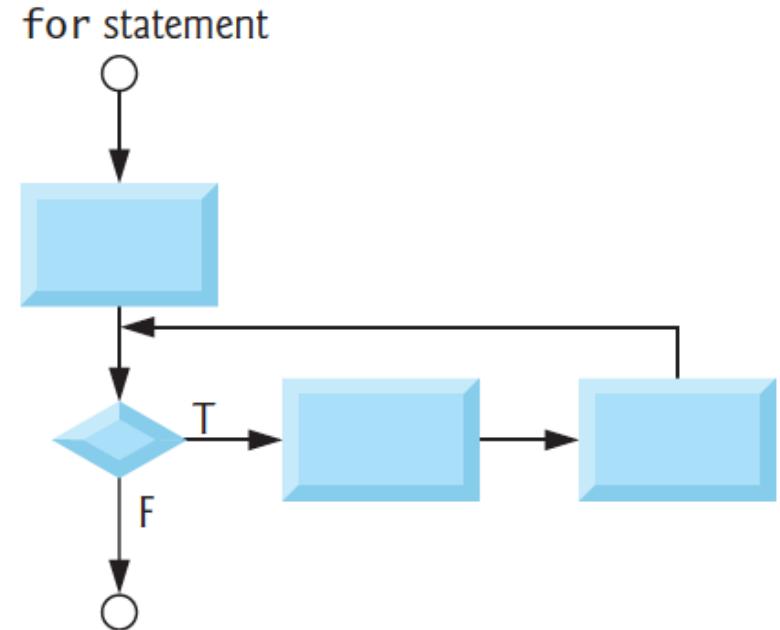
- Lo statement do...while ha un **comportamento simile al while ma esegue l'azione almeno una volta**;



Strutture di iterazione: FOR

- Lo statement `for` permette di *eseguire un'azione iterativamente fin tanto che una certa condizione viene verificata;*

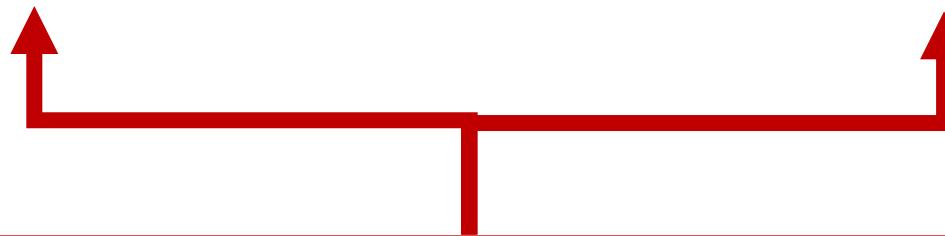
```
for(expr1; expr2; expr3){
    statement
}
```



Strutture di iterazione: FOR

- Lo statement *for* permette di *eseguire un'azione iterativamente fin tanto che una certa condizione viene verificata*;

```
expr1;  
while(expr2) {  
    statement  
for(expr1; expr2; expr3){  
        statement  
}                                }  
    expr3;
```



SONO SEMANTICAMENTE EQUIVALENTI

La variabili flag

- La variabile di *flag* concettualmente può assumere solo due valori:
 - *vero o falso*;
 - *1 o 0*;
 - *acceso o spento*;
 -
- Viene spesso utilizzata per *segnalare* se:
 - un dato evento si è verificato oppure no;
 - il sistema si trova in un certo stato oppure no;
- Per verificare che il sistema si trova in un certo stato vengono fatti dei controlli riguardo al valore della variabile di *flag*;

La variabili flag

- Una variabile booleana può essere definita in due modi:
 - in *ANSI C* il tipo booleano non esiste e di conseguenza è possibile emulare il comportamento del valore booleano dichiarando una **variabile intera** che verrà inizializzata con **0** o **1**:
int flag = 0;
oppure
int flag = 1;
 - in *C99* hanno introdotto il tipo booleano e la relativa sintassi è la seguente:
_Bool flag = 0;
Oppure
_Bool flag = 1;

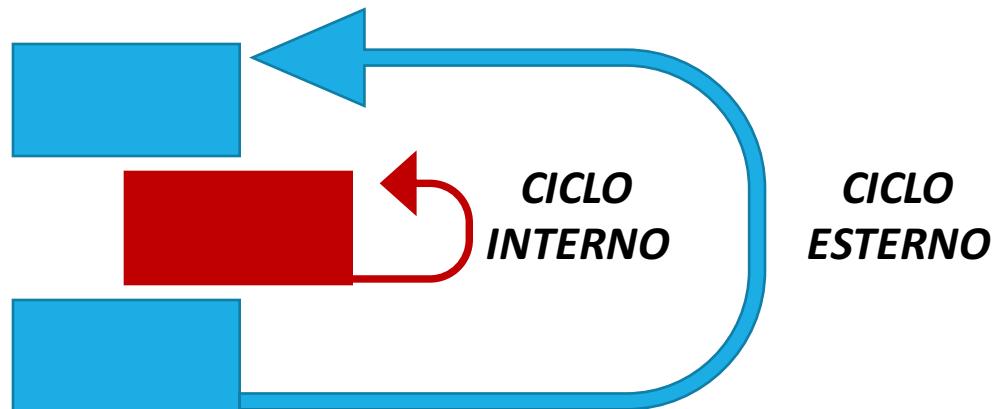
La variabili flag

- Per capire meglio il concetto si faccia riferimento al seguente frammento di codice:

```
1      i f ( number%2==0){  
2          flag=1;  
3      }  
4  
5      i f ( flag ){  
6          printf( " Pari " );  
7      } e l s e {  
8          printf( " Dispari " );  
9      }
```

Cicli annidati

- Il C permette di annidare inserire cicli all'intero di altri cicli.
- Questo processo prende il nome di ***annidamento***;
- Ogni volta che il ciclo esterno si ripete, quelli interni:
 - ricominciano daccapo;
 - rivalutano tutte le componenti di controllo;
 - rieseguono tutte le istruzioni;



Cicli annidati

- Come esempio si prenda in considerazione il seguente segmento di programma:

```
for (x=1; x<=X_ultimo; x++){  
    for (y=1; y<=Y_ultimo; y++){  
        prodotto = x * y;  
        printf (' 'x * y = %d \n ' ', prodotto);  
    }  
}
```

- Cosa stampa il seguente frammento di codice?
- Quante ripetizioni vengono svolte in totale?

Cicli annidati

```
for (x=1; x<=X_ultimo; x++){
    for (y=1; y<=Y_ultimo; y++){
        prodotto = x * y;
        printf (''x * y = %d \n '' , prodotto);
    }
}
```

- Cosa stampa il seguente frammento di codice?

1 * **1** = **1**

1 * **2** = **2**

1 * **3** = **3**

2 * **1** = **2**

2 * **2** = **4**

2 * **3** = **6**

...

Cicli annidati

```
for (x=1; x<=X_ultimo; x++){
    for (y=1; y<=Y_ultimo; y++){
        prodotto = x * y;
        printf (''x * y = %d \n '' , prodotto);
    }
}
```

- Quante ripetizioni vengono svolte in totale?

X_ultimo = 5;

x = 1 -> *y* = 1

Y_ultimo = 3;

y = 2

y = 3

x = 2 -> *y* = 1

y = 2

y = 3

...

Cicli annidati

```
for (x=1; x<=X_ultimo; x++){
    for (y=1; y<=Y_ultimo; y++){
        prodotto = x * y;
        printf (''x * y = %d \n'', prodotto);
    }
}
```

- Quante ripetizioni vengono svolte in totale?

$X_ultimo = 5;$

$Y_ultimo = 3;$

$x = 1 \rightarrow y = 1$

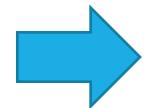
$y = 2$

$y = 3$

$x = 2 \rightarrow y = 1$

$y = 2$

$y = 3$



$X_ultimo * Y_ultimo$

...

ESERCIZI

Programmazione strutturata in C

FONDAMENTI DI PROGRAMMAZIONE

Esercizio 1



15 min

Dire cosa fa il seguente programma C:



```
5     int n, m, val, product;
6     int flag = 0;
7
8     printf("Inserire il primo numero\n");
9     scanf("%d", &n);
10
11    printf("Inserire il secondo numero\n");
12    scanf("%d", &m);
13
14    product = n*m;
15
16    if (n > m)
17        val = n;
18    else
19        val = m;
20
21    while (val < product && flag == 0) {
22        if (val%n == 0 && val%m == 0)
23            flag = 1;
24        else
25            val = val+1;
26    }
27    printf("Val: %d\n", val);
```

Esercizio 2



20 min

Definire e implementare un programma C che chieda all'utente di inserire un numero intero positivo N e determini il massimo intero M tale che la somma dei primi M interi sia minore o uguale a N.

Esempio. Se $N=15$ allora il programma dovrà restituire $M = 5$, poiché

$$1 + 2 + 3 + 4 + 5 = 15$$

mentre

$$1 + 2 + 3 + 4 + 5 + 6 = 21$$

Esercizio 3



20 min

Definire e implementare un programma C che riceva in ingresso una sequenza di interi positivi terminata da uno zero e produca come risultato la media intera dei valori di ingresso, escluso lo zero.

Ogni volta che viene inserito un numero, il programma deve controllare che sia maggiore o uguale a zero e, nel caso non lo sia, deve avvisare l'utente dell'errore e riacquisire un nuovo numero.

*Implementare l'algoritmo facendo uso del **costrutto do-while**.*

Esercizio 4



15 min

Il triangolo di Floyd è un triangolo rettangolo costruito riempendo le righe del triangolo con numeri naturali consecutivi, partendo da 1 nell'angolo in alto a sinistra.

1				
2	3			
4	5	6		
7	8	9	10	
11	12	13	14	15

Definire e implementare un programma C che chieda all'utente di inserire un numero intero N e visualizzi le prime N righe del triangolo di Floyd.

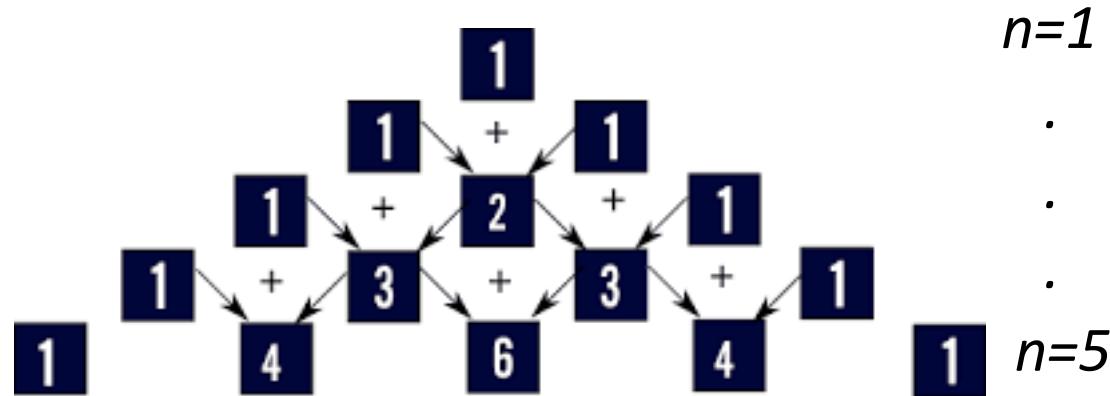
Esercizio 5



20 min

Il triangolo di Tartaglia è una tabella composta da numeri naturali in cui i coefficienti della riga n -esima corrispondono ai coefficienti dello sviluppo della potenza del binomio $(a + b)^{n-1}$.

Ogni elemento interno si ottiene dalla somma dei due numeri della riga precedente che stanno sopra di esso.



Definire e implementare un programma C che chieda all'utente di inserire un numero intero n maggiore o uguale a 1 e visualizzi le prime n righe del triangolo di Tartaglia.

Esercizio 6



20 min

Dire cosa fa il seguente programma C:

```
1 #include<stdio.h>
2
3 int main(void){
4
5     int i=0;
6     int p,n,r,q;
7
8     printf("Inserire il valore per n e p: \n");
9     scanf("%d %d", &n, &p);
10
11    if(n>=0){
12        while(1){
13            if(n>i*p && n<=(i+1)*p)
14                break;
15            i++;
16        }
17    }else{
18        while(1){
19            if(n>i*p && n<=(i+1)*p)
20                break;
21            i--;
22        }
23    }
25    q = i ;
26    r = n - q * p ;
27
28    printf("q = %d \n", q);
29    printf("r = %d \n", r);
30
31    return 0;
32 }
```

HOMEWORKS

FONDAMENTI DI PROGRAMMAZIONE

HOMEWORKS



Dire cosa fa il seguente programma C:

```
1 #include <stdio.h>
2
3 int main(void){
4
5     int i=0, count=0, result = 0;
6     int integer, flag;
7
8     printf("Inserire un intero: \n");
9     scanf("%d", &integer);
10
11    for (i = 0; i <= integer; ++i){
12        count++;
13        if(i == count){
14            flag = 1;
15            continue;
16        } else {
17            flag = 0;
18            break;
19        }
20
21        if(flag){
22            result = result + count;
23        } else{
24            result = result - count;
25        }
26    }
27
28    printf("Result: %d\n", result + 1);
29
30    return 0;
31 }
```

HOMEWORKS



Definire e implementare un programma C che:

1. *chieda di inserire all'utente una serie di numeri lunga **almeno** quattro cifre fin tanto che non viene inserito il valore zero;*
2. *stampi la somma degli ultimi tre numeri inseriti, escluso lo 0.*

HOMEWORKS



Definire e implementare un programma C che controlli se un carattere inserito dall’utente è una lettera dell’alfabeto, una cifra o un carattere speciale.

*Il programma dovrà essere **non-case sensitive**, ovvero se l’utente inserisce il carattere ‘a’ oppure ‘A’, l’output dovrà essere lo stesso.*

Esempio. Se *character*=‘a’ allora il programma dovrà restituire “Lettera dell’alfabeto.”

Se *character* = ‘A’ allora il programma dovrà restituire “Lettera dell’alfabeto.”

Implementare due versioni del programma:

- 1. Utilizzare i caratteri (es. ‘a’);*
- 2. Utilizzare la codifica ascii;*

HOMEWORKS



Definire e implementare un programma C che chieda di inserire all’utente due valori interi, indicanti il numero di righe e colonne di una tabella.

La tabella dovrà essere stampata nel modo seguente:

Esempio. N_righe = 5 e N_colonne = 5;

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

HOMEWORKS



Un numero naturale è primo se è maggiore di 1 e se è divisibile per 1 e per sé stesso.

Definire e implementare un programma C che chieda all'utente di inserire un numero naturale n e faccia il prodotto dei numeri primi all'interno dell'intervallo che va da 0 a n .

Implementare due versioni del programma:

- 1. deve fare uso delle **variabili di flag**;*
- 2. deve fare uso dell'**istruzione speciale break**;*