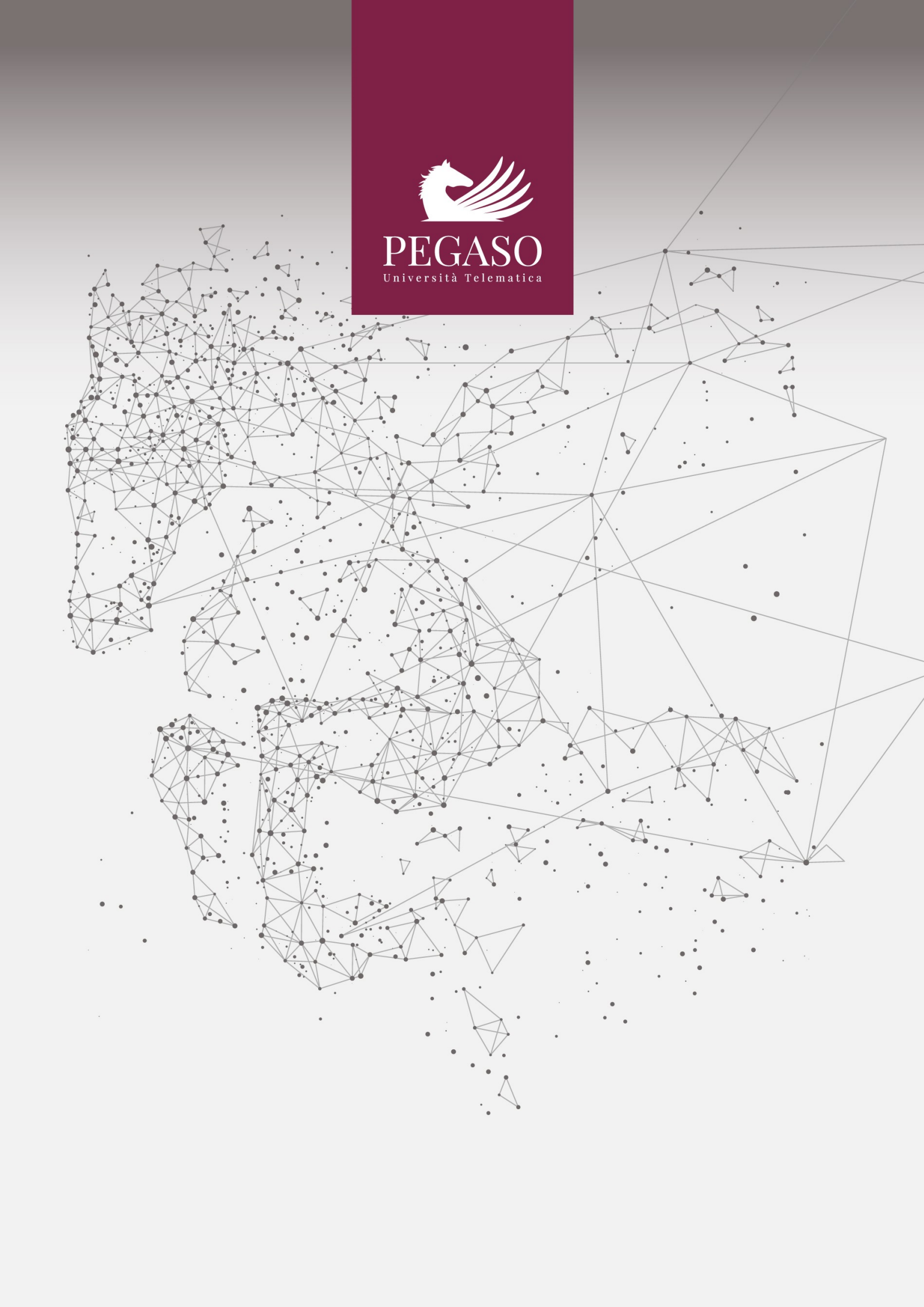
PEGASO
Università Telematica

Indice

1. Esempio di un gioco d'azzardo: introduzione di enum

Un popolare gioco d'azzardo è un gioco di dadi noto come "craps".

Le regole del gioco sono semplici:

Un giocatore lancia due dadi. Ogni dado ha sei facce.

Queste facce contengono 1, 2, 3, 4, 5 e 6 pallini.

Quando i dadi si fermano, si calcola la somma dei pallini sulle due facce superiori.

Se al primo lancio la somma è 7 o 11, il giocatore vince.

Se al primo lancio la somma è 2, 3 o 12 (chiamata "craps"), il giocatore perde (cioè vince il banco).

Se al primo lancio la somma è 4, 5, 6, 8, 9 o 10, quella somma diventa il "punteggio" del giocatore.

Per vincere si devono continuare a lanciare i dadi finché si "fa il proprio punteggio".

Il giocatore perde se ottiene come somma 7 prima di fare il proprio punteggio.

Il seguente programma realizza il gioco del craps.

```
01  // Programma in C
02  // Simulazione del gioco del craps.
03  #include <stdio.h>
04  #include <stdlib.h>
05  #include <time.h> // prototipo per la funzione time
06
07  // le costanti di enumerazione indicano lo stato del gioco
08  enum Status { CONTINUE, WON, LOST };
09
10  int rollDice(void); // prototipo di funzione
11
12  int main(void)
13  {
```

```
14      //  randomizza il generatore di numeri casuali
15      srand(time(NULL));
16
17      int myPoint; // punteggio per vincere
18      enum Status gameStatus; // conterra' CONTINUE, WON o LOST
19      int sum; = rollDice(); // primo lancio dei dadi
20
21      // determina lo stato del gioco in base alla somma dei dadi
22      switch(sum) {
23
24         // si vince al primo lancio
25         case 7: // si vince con 7
26         case 11: // si vince con 11
27            gameStatus = WON;
28            break;
29
30         // si perde al primo lancio
31         case 2: // si perde con 2
32         case 3: // si perde con 3
33         case 12: // si perde con 12
34            gameStatus = LOST;
35            break;
36
37         // ricorda il punteggio
38         default:
39            gameStatus = CONTINUE; // continua a lanciare
40            myPoint = sum; // ricorda il punteggio
41            printf("Point is %d\n", myPoint);
42            break; // opzionale
43      }
44
45      // finche' il gioco non si conclude
```

```
46      while (CONTINUE == gameStatus) { //  il gioco continua
47         sum = rollDice(); // lancia di nuovo i dadi
48
49         // determina lo stato del gioco
50         if (sum == myPoint) { // si vince facendo il punteggio
51            gameStatus = WON;
52         }
53         else {
54            if (7 == sum) { // si perde lanciando il 7
55               gameStatus = LOST;
56            }
57         }
58      }
59
60      // stampa il messaggio di vincita o di perdita
61      if (WON == gameStatus) { // il giocatore ha vinto?
62         puts("Player wins");
63      }
64      else { // il giocatore ha perso
65         puts("Player loses");
66      }
67   }
68
69   // lancia i dadi, calcola la somma e stampa i risultati
70   int rollDice(void)
71   {
72      int die1 = 1 + (rand() % 6); // valore primo dado
73      int die2 = 1 + (rand() % 6); // valore secondo dado
74
75      // stampa i risultati di questo lancio
76      printf("Player rolled %d+%d = %d\n", die1,die2,die1+die2);
77      return die1 + die2; // restituisci la somma dei dadi
```

```
78  }
```

Esempi di esecuzione del programma - gioco del craps.

Il giocatore vince al primo lancio

```
Player rolled 5 + 6 = 11

Player wins
```

Il giocatore vince a un lancio successivo

```
Player rolled 4 + 1 = 5

Point is 5

Player rolled 6 + 2 = 8

Player rolled 2 + 1 = 3

Player rolled 3 + 2 = 5

Player wins
```

Il giocatore perde al primo lancio

```
Player rolled 1 + 1 = 2

Player loses
```

Il giocatore perde a un lancio successivo

```
Player rolled 6 + 4 = 10

Point is 10

Player rolled 3 + 4 = 7

Player loses
```

Nelle regole del gioco notate che il giocatore deve lanciare due dadi al primo lancio, e deve fare così in seguito in tutti i lanci successivi.

Definiamo una funzione `rollDice` per simulare il lancio dei dadi e calcolare e stampare la loro somma.

La funzione `rollDice` è definita una sola volta, ma è chiamata in due punti diversi nel programma (righe 19 e 47). È interessante notare che `rollDice` non riceve alcun argomento, così abbiamo inserito void nella lista dei parametri (riga 70).

La funzione `rollDice` restituisce la somma dei due dadi, così nella sua intestazione e nel suo prototipo di funzione è indicato il tipo di ritorno `int`.

Enumerazioni

Il gioco è ragionevolmente articolato.

Il giocatore può vincere o perdere al primo lancio, oppure vincere o perdere a un qualunque lancio successivo.

La variabile gameStatus di un nuovo tipo – enum Status – memorizza lo stato corrente.

La riga 8 crea un tipo definito dal programmatore, chiamato enumerazione.

Un'enumerazione, introdotta dalla parola chiave enum, è un insieme di costanti intere rappresentate da identificatori.

Le costanti di enumerazione aiutano a rendere i programmi più leggibili e più facili da mantenere.

I valori in un `enum` iniziano con 0 e sono incrementati di 1.

Nella riga 8 la costante `CONTINUE` ha il valore 0, `WON` ha il valore 1 e `LOST` ha il

valore 2.

È anche possibile assegnare un valore intero a ciascun identificatore in un `enum`.

Gli identificatori in un'enumerazione devono essere unici, ma i valori possono essere duplicati.

☹ *Errore comune di programmazione*

Assegnare un valore a una costante di enumerazione dopo che è stata definita è un errore di sintassi.

😊 *Buona pratica di programmazione*

Usate solo lettere maiuscole nei nomi delle costanti di enumerazione per far sì che queste costanti risaltino in un programma e per indicare che non sono delle variabili.

Quando si vince nel gioco, o al primo lancio o a un lancio successivo, `gameStatus` assume il valore `WON`.

Quando si perde, o al primo lancio o a un lancio successivo, `gameStatus` assume il valore `LOST`.

Diversamente, `gameStatus` assume il valore `CONTINUE` e il gioco continua.

Fine del gioco al primo lancio

Dopo il primo lancio, se il gioco è finito, l'istruzione `while` (righe 46–58) viene saltata perché `gameStatus` non ha il valore `CONTINUE`.

Il programma procede con l'istruzione `if...else` alle righe 61–66, che stampa "Player wins" se `gameStatus` è `WON`, altrimenti stampa "Player loses".

Fine del gioco a un lancio successivo

Dopo il primo lancio, se il gioco non è finito, il valore di sum viene salvato in `myPoint`. L'esecuzione procede con l'istruzione `while` perché `gameStatus` ha il valore `CONTINUE`.

A ogni iterazione del `while`, `rollDice` è invocata per produrre un nuovo valore di `sum`.

Se `sum` coincide con `myPoint`, `gameStatus` assume il valore `WON` per indicare che il giocatore ha vinto, il test del `while` fallisce, l'istruzione `if...else` stampa "Player wins" e l'esecuzione termina.

Se `sum` è uguale a 7 (riga 54), `gameStatus` assume il valore `LOST` per indicare che il giocatore ha perso, il test del `while` fallisce, l'istruzione `if...else` stampa "Player loses" e l'esecuzione termina.

Architettura di controllo

Notate l'interessante architettura di controllo del programma.

Abbiamo usato due funzioni – `main` e `rollDice` – e le istruzioni `switch`, `while`, `if...else` annidati e `if` annidati.

Un'enumerazione, introdotta dalla parola chiave `enum`, è un insieme di costanti intere di enumerazione rappresentate da identificatori.

I valori in un `enum` partono da 0 e sono incrementati nell'ordine di 1.

In questo esempio di `enum months`, gli identificatori sono quindi impostati con gli interi da 0 a 11.

```
enum months {
   JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
};
```

Per numerare i mesi da 1 a 12, usate la seguente enumerazione:

```
enum months {
   JAN=1, FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
};
```

Poiché il primo valore nella enumerazione è esplicitamente posto a 1, i valori restanti sono incrementati a partire da 1, ottenendo così i valori da 1 a 12

Ricordate che:

- Gli identificatori in un'enumerazione devono essere unici
- Il valore di ogni costante in un'enumerazione può essere impostato esplicitamente nella definizione, assegnando un valore all'identificatore
- Più membri di un'enumerazione possono avere lo stesso valore costante

Esercizio

1. Determinate l'output del seguente programma:

```
#include <stdio.h>
enum week { MON, TUE, WED };
int main(void)
{
   enum week day;
   day = WED;
   printf( "%d", day);
}
```

Output: 2

2. Determinate l'output del seguente programma:

```
#include <stdio.h>
enum week { MON, TUE, WED, THU, FRI, SAT, SUN };
int main(void)
{
   int i;
```

```
    for( i=MON; i<=SUN; i++) {
        printf( "%d ", i);
    }
}
```

Output: 0, 1, 2, 3, 4, 5, 6

3. Proposta di esercizio

Modificate il programma del gioco craps per consentire di scommettere

Impacchettate come funzione la porzione del programma che esegue un giro del gioco del craps

Inizializzate la variabile `bankBalance` a 1000 dollari

Richiedete al giocatore di inserire una scommessa come valore per la variabile `bet`

Usate un ciclo `while` per controllare che `bet` sia minore o uguale a `bankBalance` e, se non lo è, richiedete all'utente di reinserire il valore di `bet` finché non viene inserito un valore valido

Dopo l'inserimento di un valore valido, fate eseguire un giro del gioco del craps

Se il giocatore vince:

- aumentate il valore di `bankBalance` della quantità pari al valore di `bet`
- e stampate il nuovo `bankBalance`

Se il giocatore perde:

- diminuite il valore di `bankBalance` della quantità pari al valore di `bet`,
- stampate il nuovo `bankBalance`
- e controllate se `bankBalance` è diventato zero e, se è così, stampate il messaggio "Sorry, lost!"

```
// Soluzione
#include <stdio.h>
#include <stdlib.h>
```

```
#include <time.h>

// le costanti di enumerazione rappresentano lo stato del gioco
enum Status {CONTINUE, WON, LOST};
int rollDice(void); // prototipo di funzione
enum Status craps(void); // prototipo di funzione
void chatter(void); // prototipo di funzione

int main()
{
   srand(time(NULL)); // seme per il generatore di numeri casuali
   // stampa il saldo corrente e richiedi la puntata
   int bankBalance = 1000; // saldo del banco corrente
   printf("You have $%d in the bank.\n", bankBalance);
   printf("Place your wager: ");
   int wager; // puntata per il giro corrente
   scanf("%d", &wager);
   // ripeti finche' la puntata non e' valida
   while(wager <= 0 || wager > 1000) {
      printf("Please bet a valid amount.\n");
      scanf("%d", &wager);
   }
   enum Status result = craps(); // gioca il giro di craps
   // se il giocatore ha perso il giro corrente
   if (LOST == result) {
      // diminuisci il saldo in base alla puntata e stampa quello nuovo
      bankBalance -= wager;
      printf("Your new bank balance is $%d\n", bankBalance);
      // se il saldo e' 0
      if (0 == bankBalance) {
         printf("Sorry. You are Busted! Thank You For Playing.\n");
      }
   }
   else { // il giocatore ha vinto la partita
      // aumenta il saldo in base alla puntata e stampa quello nuovo
      bankBalance += wager;
      printf("Your new bank balance is $%d\n", bankBalance);
   }
}
```

```
// lancio dei dadi, calcola la somma e stampa i risultati
int rollDice(void)
{
   int die1 = 1 + rand() % 6; // genera il valore casuale die1
   int die2 = 1 + rand() % 6; // genera il valore casuale die2
   int workSum = die1 + die2; // somma die1 e die2
   // stampa i risultati di questo lancio
   printf("Player rolled %d + %d = %d\n", die1, die2, workSum);
   return workSum; // restituzione della somma dei dadi
}

// craps gioca un giro di craps, restituisce il risultato del giro
enum Status craps(void)
{
   enum Status gameStatus; // puo' contenere CONTINUE, WON o LOST
   int myPoint; // valore del punto
   int sum = rollDice(); // primo lancio di dadi
   // determina stato del gioco e punto in base alla somma dei dadi
   switch (sum) {
      // vince al primo lancio
      case 7:
      case 11:
         gameStatus = WON;
         chatter();
         break; // uscita dallo switch
      // perde al primo lancio
      case 2:
      case 3:
      case 12:
         gameStatus = LOST;
         chatter();
         break; // uscita dallo switch
      // ricorda il punto
      default:
         gameStatus = CONTINUE;
         myPoint = sum;
         printf("Point is %d\n", myPoint);
         chatter();
```

```
            break; // uscita dallo switch
      }
      // finche' il gioco non e' completo
      while (CONTINUE == gameStatus) {
         chatter();
         sum = rollDice(); // nuovo lancio di dadi

         // determina lo stato del gioco
         if (sum == myPoint) {
            gameStatus = WON; // vittoria facendo punto
         }
         else {
            if (7 == sum) {
               gameStatus = LOST; // sconfitta tirando 7
            }
         }
      }
      // stampa il messaggio di vittoria o sconfitta e restituisce lo stato
      if (WON == gameStatus) {
         printf("Player wins\n");
         return WON;
      }
      else {
         printf("Player loses\n");
         return LOST;
      }
}

// chatter stampa i messaggi a caso
void chatter(void)
{
      int select = 1 + rand() % 6;
      // scelta casuale del messaggio
      switch (select) {
         case 1:
            printf("Oh, you're going for broke, huh?\n");
            break; // uscita dallo switch
         case 2:
            printf("Aw cmon, take a chance!\n");
```

```
            break; // uscita dallo switch
        case 3:
            printf("Hey, I think this guy is going to break the bank!!\n");
            break; // uscita dallo switch
        case 4:
            printf("You're up big. Now's the time to cash!\n");
            break; // uscita dallo switch
        case 5:
            printf("Way too lucky! Those dice have to be loaded!\n");
            break; // uscita dallo switch
        case 6:
            printf("Bet it all! Bet it all!\n");
            break; // uscita dallo switch
        default:
            break; // uscita dallo switch
    }
}
```

Riferimenti bibliografici

- Paul Deitel, Harvey Deitel, "Il linguaggio C – Fondamenti e tecniche di programmazione", Libro edito da Pearson Italia. Include anche utili esercizi di autovalutazione.