Funzioni
Prima Parte

9/10/2024

Per ognuno dei seguenti set di interi, scrivere un un'unica linea di codice che produce un numero random da quell'insieme:

a) 2,4,6,8,10
b) 3,5,7,9,11
c) 6,10,14,18,22

Per ognuno dei seguenti set di interi, scrivere un un'unica linea di codice che produce un numero random da quell'insieme:

a) 2,4,6,8,10
b) 3,5,7,9,11
c) 6,10,14,18,22

a) 2, 4, 6, 8, 10.
Answer: `printf("%d\n", 2 * (1 + rand() % 5));`
b) 3, 5, 7, 9, 11.
Answer: `printf("%d\n", 1 + 2 * (1 + rand() % 5));`
c) 6, 10, 14, 18, 22.
Answer: `printf("%d\n", 6 + 4 * (rand() % 5));`

a) Cosa fa il seguente pezzo di codice?
b) Cosa succede se scambiamo la linea 7 con la 8

```
1  #include <stdio.h>
2
3  int main(void) {
4     int c = '\0'; // variable to hold character input by user
5
6     if ((c = getchar()) != EOF) {
7        main();
8        printf("%c", c);
9     }
10 }
```

a) Cosa fa il seguente pezzo di codice?
b) Cosa succede se scambiamo la linea 7 con la 8

```
1  #include <stdio.h>
2
3  int main(void) {
4     int c = '\0'; // variable to hold character input by user
5
6     if ((c = getchar()) != EOF) {
7        main();
8        printf("%c", c);
9     }
10 }
```

Risposta: Immette un carattere e richiama ricorsivamente main() finché non viene inserito il carattere EOF viene inserito. Ogni carattere inserito viene quindi emesso in ordine inverso. Se si cambiano le righe 7 e 8, i caratteri vengono visualizzati nell'ordine di inserimento.
le righe 7 e 8, i caratteri vengono visualizzati nell'ordine di inserimento.

```c
#include <stdio.h>

int mystery(int a, int b); // function prototype

int main(void) {
   printf("%s", "Enter two positive integers: ");
   int x = 0; // first integer
   int y = 0; // second integer
   scanf("%d%d", &x, &y);

   printf("The result is %d\n", mystery(x, y));
}

// Parameter b must be a positive integer
// to prevent infinite recursion
int mystery(int a, int b) {
   // base case
   if (1 == b) {
      return a;
   }
   else { // recursive step
      return a + mystery(a, b - 1);
   }
}
```

Cosa fa questo pezzo di codice?

```c
#include <stdio.h>

int mystery(int a, int b); // function prototype

int main(void) {
   printf("%s", "Enter two positive integers: ");
   int x = 0; // first integer
   int y = 0; // second integer
   scanf("%d%d", &x, &y);

   printf("The result is %d\n", mystery(x, y));
}

// Parameter b must be a positive integer
// to prevent infinite recursion
int mystery(int a, int b) {
   // base case
   if (1 == b) {
      return a;
   }
   else { // recursive step
      return a + mystery(a, b - 1);
   }
}
```

Cosa fa questo pezzo di codice?

Simula la moltiplicazione, sommando a, a se stesso b volte.

5.18. ***(Pari o dispari)*** Scrivete un programma che riceva in ingresso una serie di interi e li passi uno alla volta alla funzione `isEven`, che usa l'operatore di resto per determinare se un intero è pari. La funzione deve prendere un argomento intero e restituire `1` se l'intero è pari e `0` nel caso contrario.

```c
// Exercise 5.18 Solution
#include <stdio.h>

int isEven(int a); // function prototype

int main() {
   // loop for 3 inputs
   for (int i = 1; i <= 3; ++i) {
      printf("%s", "Enter an integer: ");
      int x; // current input
      scanf("%d", &x);

      // determine if input is even
      if (isEven(x)) {
         printf("%d is an even integer\n\n", x);
      }
      else {
         printf("%d is not an even integer\n\n", x);
      }
   }
}

// isEven returns true if a is even
int isEven(int a) {
   return !(a % 2);
}
```

5.16. ***(Esponenziazione)*** Scrivete una funzione `integerPower(base, exponent)` che restituisca il valore di

$$base^{exponent}$$

Per esempio, `integerPower(3, 4) = 3 * 3 * 3 * 3`. Supponete che `exponent` sia un intero positivo diverso da zero e che `base` sia un intero. La funzione `integerPower` deve usare un'istruzione `for` per controllare il calcolo. Non usate alcuna funzione della libreria math.

```c
// Exercise 5.16 Solution
#include <stdio.h>

int integerPower(int b, int e);

int main() {
   printf("%s", "Enter integer base and exponent: ");
   int base; // integer base
   int exp; // integer exponent
   scanf("%d%d", &base, &exp);

   printf("%d to the power %d is: %d\n",
      base, exp, integerPower(base, exp));
}

// integerPower calculates and returns b raised to the e power
int integerPower(int b, int e) {
   int product = 1; // resulting product

   // multiply product times b (e repetitions)
   for (int i = 1; i <= e; ++i) {
      product *= b;
   }

   return product; // return resulting product
}
```

5.15. ***(Calcolo dell'ipotenusa)*** Definite una funzione chiamata `hypotenuse` che calcoli la lunghezza dell'ipotenusa di un triangolo rettangolo quando sono dati gli altri due lati. La funzione deve ricevere due argomenti di tipo `double` e restituire l'ipotenusa come un `double`. Testate il vostro programma con i valori dei lati specificati nella seguente tabella:

Lato 1	Lato 2
3.0	4.0
5.0	12.0
8.0	15.0

```c
// Exercise 5.15 Solution
#include <stdio.h>
#include <math.h>

double hypotenuse(double s1, double s2); // function prototype

int main() {
   // loop 3 times
   for (int i = 1; i <= 3; ++i) {
      printf("%s", "Enter the sides of the triangle: ");
      double side1; // value for first side
      double side2; // value for second side
      scanf("%lf%lf", &side1, &side2);

      // calculate and display hypotenuse value
      printf("Hypotenuse:  %.1f\n\n", hypotenuse(side1, side2));
   }
}

// hypotenuse calculates value of hypotenuse of
// a right triangle given two side values
double hypotenuse(double s1, double s2) {
   return sqrt(pow(s1, 2) + pow(s2, 2));
}
```

5.9. ***(Costo del parcheggio)*** Un garage fa pagare una tariffa minima di $2 ,00 per parcheggiare fino a tre ore, più $0 ,50 all'ora per ogni ora *o parte di essa* oltre le tre ore. Il costo massimo per un periodo di 24 ore è di $10,00. Supponete che nessuna macchina resti parcheggiata per più di 24 ore. Scrivete un programma che calcoli e stampi i costi del parcheggio per ciascuno dei tre clienti che ieri hanno parcheggiato le loro auto in questo garage. Dovete inserire le ore di parcheggio per ogni cliente. Il vostro programma deve stampare i risultati in un formato tabellare e deve calcolare e stampare il totale degli incassi di ieri. Il programma deve usare la funzione `calculateCharges` per determinare il costo per ogni cliente. Il vostro output deve avere il seguente formato:

```
Car      Hours       Charge
1          1.5         2.00
2          4.0         2.50
3         24.0        10.00
TOTAL     29.5        14.50
```

```c
// Exercise 5.9 Solution
#include <stdio.h>
#include <math.h>

double calculateCharges(double hours); // function prototype

int main() {
   printf("%s", "Enter the hours parked for 3 cars: ");
   int first = 1; // flag for printing table headers
   double totalCharges = 0.0; // total charges for all cars
   double totalHours = 0.0; // total number of hours for all cars

   // loop 3 times for 3 cars
   for (int i = 1; i <= 3; ++i) {
      double h; // number of hours for current car
      scanf("%lf", &h);
      totalHours += h; // add current hours to total hours

      // if first time through loop, display headers
      if (first) {
         printf("%5s%15s%15s\n", "Car", "Hours", "Charge");

         // set flag to false to prevent from printing again
         first = 0;
      }

      // calculate current car's charge and update total
      double currentCharge; // parking charge for current car
      totalCharges += (currentCharge = calculateCharges(h));

      // display row data for current car
      printf("%5d%15.1f%15.2f\n", i, h, currentCharge);
   }

   // display row data for totals
   printf("%5s%15.1f%15.2f\n", "TOTAL", totalHours, totalCharges);
}

// calculateCharges returns charge according to number of hours
double calculateCharges(double hours) {
   double charge; // calculated charge

   // $2 for up to 3 hours
   if (hours < 3.0) {
      charge = 2.00;
   }

   // $.50 for each hour or part thereof in excess of 3 hours
   else if (hours < 19.0) {
      charge = 2.00 + .50 * ceil(hours - 3.0);
   }

   else { // maximum charge $10
      charge = 10.0;
   }

   return charge; // return calculated charge
}
```

5.31. ***(Lancio di una moneta)*** Scrivete un programma che simuli il lancio di una moneta. Per ogni lancio della moneta il programma deve stampare `Heads` (testa) o `Tails` (croce). Fate lanciare al programma la moneta 100 volte e contate il numero di volte in cui compare ogni lato della moneta. Stampate i risultati. Il programma deve chiamare una funzione separata `flip` che non riceve alcun argomento e restituisce `0` per croce e `1` per testa. Se il programma simula in maniera realistica il lancio della moneta, allora ogni lato della moneta deve comparire approssimativamente la metà delle volte, per un totale di approssimativamente 50 teste e 50 croci.

```c
// Exercise 5.31 Solution
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int flip(); // function prototype

int main() {
   srand(time(NULL)); // seed random number generator

   int headCount = 0; // total Heads count
   int tailCount = 0; // total Tails count

   // simulate coin toss 100 times
   for (int loop = 1; loop <= 100; ++loop) {
      // simulate coin toss, 0 refers to tails
      if (flip() == 0) {
         ++tailCount; // update Tails count
      }
      else {
         ++headCount; // update Heads count
      }

      if (loop % 10 == 0) {
         puts("");
      }
   }

   // display totals
   printf("\nThe total number of Heads was %d\n", headCount);
   printf("The total number of Tails was %d\n", tailCount);
}

// flip uses random number to simulate coin toss
int flip() {
   int value = rand() % 2; // 0 is tails, 1 is heads

   // display result of flip
   if (0 == value) {
      printf("%s", "Tails ");
   }
   else {
      printf("%s", "Heads ");
   }

   return value; // return result of coin toss
}
```

5.32. ***(Indovina il numero)*** Scrivete un programma in C che realizzi il gioco “indovina il numero” come segue: il vostro programma sceglie il numero da indovinare selezionando a caso un intero nell’intervallo da 1 a 1. 000. Il programma quindi stampa:

```
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
```

Il giocatore allora scrive una prima risposta. Il programma risponde con una delle seguenti frasi:

```
1. Excellent! You guessed the number!
   Would you like to play again (y or n)?
2. Too low. Try again.
3. Too high. Try again.
```

Se la risposta del giocatore è sbagliata, il vostro programma deve entrare in un ciclo finché il giocatore non indovina finalmente il numero giusto. Deve continuare a dire al giocatore `Too high` o `Too low` per aiutarlo a “convergere” sulla risposta corretta.

```c
// Exercise 5.32 solution
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void guessGame(void); // function prototype

int main() {
   srand(time(NULL)); // seed random number generator
   guessGame();
}

// guessGame generates numbers between 1 and 1000
// and checks user's guess
void guessGame(void) {
   int response; // response to continue game

   // loop until user quits game
   do {
      // generate random number between 1 and 1000
      // 1 is shift, 1000 is scaling factor
      int x = 1 + rand() % 1000;

      // prompt for guess
      puts("\nI have a number between 1 and 1000.");
      puts("Can you guess my number?");
      printf("%s", "Please type your first guess.\n? ");
      int guess; // user's guess
      scanf("%d", &guess);

      // loop until correct number
      while (guess != x) {

         // if guess is too low
         if (guess < x) {
            printf("%s", "Too low. Try again.\n? ");
         }
         else { // guess is too high
            printf("%s", "Too high. Try again.\n? ");
         }

         scanf("%d", &guess);
      }

      // prompt for another game
      puts("\nExcellent! You guessed the number!");
      printf("%s", "Would you like to play again (y or n)? ");
      getchar(); // discard newline from last entered value
      response = getchar(); // response to continue game
   } while ('y' == response); // end do...while
}
```

5.19. ***(Quadrato di asterischi)*** Scrivete una funzione che stampi un quadrato di asterischi pieno il cui lato è specificato nel parametro intero `side`. Per esempio, se `side` è `4`, la funzione stampa:

```
****
****
****
****
```

```c
// Exercise 5.19 Solution
#include <stdio.h>

void square(int side); // function prototype

int main() {
   printf("%s", "Enter side: ");
   int s; // input side length
   scanf("%d", &s);

   square(s); // display solid square of asterisks
}

// square displays solid square of asterisks with specified side
void square(int side) {
   // loop side times for number of rows
   for (int i = 1; i <= side; ++i) {
      // loop side times for number of columns
      for (int j = 1; j <= side; ++j) {
         printf("%s", "*");
      }

      puts("");
   }
}
```

5.21. ***(Progetto: disegnare forme e caratteri)*** Usate tecniche simili a quelle sviluppate negli Esercizi 5. 19 e 5. 20 per produrre un programma che disegni una vasta gamma di forme.

Noi in particolare faremo il diamante