A.A. 08/09

Fondamenti di Programmazione

(canale E-O)

Docente: Prof.ssa Tiziana Calamoneri
calamo@di.uniroma1.it

Esercitatore: Dott. Roberto Petroccia
petroccia@di.uniroma1.it

Pagina del corso:
http://twiki.di.uniroma1.it/twiki/view/Programmazione1/EO/WebHome

---

Esercitazione del 15/10/08

Homework

1. Leggete con attenzione le modalità di consegna (estensioni dei file .c, formato di input ed output, etc ...).
2. Leggete con attenzione le tracce degli esercizi.
3. Consultate la pagina delle domande.
4. Non rispondete a domande di altri studenti (Controllate che le risposte provengano da docente o esercitatore del corso).
5. Compilate ed eseguite i vostri programmi su macchine Unix prima di consegnare per essere sicuri che vada tutto bene
6. NON COPIATE.

Funzioni in C

Ogni funzione è della forma:

Tipo di ritorno Nome funzione Parametri

```
int    maximum    (int x, int y) {

        CORPO DELLA FUNZIONE

}
```

Funzioni in C

Ogni funzione è della forma:

Tipo di ritorno Nome funzione Parametri

```
int    maximum    (int x, int y) {
    if (x > y) {
        return x;
    }
    return y;
}
```

Deve essere specificato l'output della funzione

Funzioni in C

Ogni funzione è della forma:

Nessun tipo di ritorno

Nome funzione

Parametri

```
void    stampaCoppia    (int x, int y) {
        printf (“x = %d - y = %d\n”, x, y);
}
```

Non c’è nessun output

Funzioni in C

- A cosa serve la segnatura?

```
#include <stdio.h>
int maximum (int x, int y);
int main() {
        int a =0, b = 0;
        printf("Inserisci primo valore:\n");
        scanf("%d", &a);
        printf("Inserisci secondo valore:\n");
        scanf("%d", &b);
        printf("Il valore massimo è %d\n", maximum(a,b));
}

int    maximum    (int x, int y) {
          if (x > y) {
              return x;
          }
          return y;
}
```

Serve a dire che esiste una funzione maximum che restituisce un intero e prende come parametri due interi se tale funzione non è definita al momento del suo uso

Funzioni in C

- A cosa serve la segnatura?

Non c'è bisogno della segnatura

```
#include <stdio.h>
int  maximum (int x, int y) {
        if (x > y) {
            return x;
        }
        return y;
}

int main() {
        int a =0, b = 0;
        printf("Inserisci primo valore:\n");
        scanf("%d", &a);
        printf("Inserisci secondo valore:\n");
        scanf("%d", &b);
        printf("Il valore massimo è %d\n", maximum(a,b));
}
```

Perché usare le funzioni?

- Permettono di scrivere del codice più chiaro
- Permettono di riutilizzare pezzi di codice senza dover riscrivere più volte la stessa cosa
- È più semplice scrivere codice per problemi più piccoli invece di scrivere il tutto in un unico blocco

Esempio uso di funzioni

- Scrivere un programma che continui a leggere un intero n da input e ne stampi il fattoriale finché n > 0

```
LEGGI intero n
FINCHÉ (n > 0)
    STAMPA fattoriale di n
    LEGGI intero n
ESCI
```

Esempio uso di funzioni

- Scrivere un programma che continui a leggere un intero n da input e ne stampi il fattoriale finché n > 0

```
#include <stdio.h>

int fattoriale (int x);

int main() {
        int n;
        scanf (“%d”, &n);
        while (n > 0) {
            printf(“ --> %d\n”, fattoriale(n));
            scanf (“%d”, &n);
        }
        return 0;
}
```

```
int fattoriale(int x) {
        int y = 0;
        if (x < 0) {
            printf(“ERROR\n”);
        }
        if (x <= 1) {
            return 1;
        }
        y = 1;
        while (x > 1) {
            y *= x--;
        }
        return y;
}
```

DOMANDE ???

Esercizi

1. Scrivere un progamma che letto un intero n da input stampi la cornice di un quadrato di lato n utilizzando il simbolo *
2. Scrivere un progamma che letti due interi m ed n stampi la cornice di un rettangolo m righe ed n colonne utilizzando il simbolo *
3. Scrivere un progamma che letto un intero n da input stampi la cornice e le diagonali di un quadrato di lato n utilizzando il simbolo *
4. Un numero si dice perfetto quando è uguale alla somma di tutti i suoi divisori escluso se stesso. Ad esempio, il numero 28, divisibile per 1, 2, 4, 7, 14 è un numero perfetto (28 = 1 + 2 + 4 + 7 + 14): lo stesso per 6 che è divisibile per 1, 2 e 3 (6 = 1 + 2 + 3). Scrivere una funzione che preso in input un numero positivo n stampi 1 se n è un numero perfetto, 0 altrimenti.

Soluzione Ex. 1

```
#include <stdio.h>

void stampaQuadrato (char c, int x);

int main() {
        int n;
        scanf (“%d”, &n);
        stampaQuadrato(‘*’, n);
        return 0;
}
```

```
void stampaQuadrato (char c, int x) {
        int i, j;
        for (i = 0; i < x; i++) {
            for (j = 0; j < x; j++) {
                if (i == 0 || i == x-1) {
                    printf(“%c”, c);
                }
                else {
                    if (j == 0 || j == x-1) {
                        printf(“%c”, c);
                    }
                    else {
                        printf(“ ”);
                    }
                }
            }
            printf(“\n”);
        }
    }
```

Soluzione Ex. 2

```
#include <stdio.h>

void stampaRettangolo (char c, int row, int col);

int main() {
        int m, n;
        scanf (“%d”, &m);
        scanf (“%d”, &n);
        stampaRettangolo(‘*’, m, n);
        return 0;
}
```

```
void stampaRettangolo (char c, int row, int col){
        int i, j;
        for (i = 0; i < row; i++) {
            for (j = 0; j < col; j++) {
                if (i == 0 || i == row-1) {
                    printf(“%c”, c);
                }
                else {
                    if (j == 0 || j == col-1) {
                        printf(“%c”, c);
                    }
                    else {
                        printf(“ ”);
                    }
                }
            }
            printf(“\n”);
        }
}
```

Soluzione Ex. 2

```
#include <stdio.h>

void stampaRettangolo (char c, int row, int col);

int main() {
        int m, n;
        scanf (“%d”, &m);
        scanf (“%d”, &n);
        stampaRettangolo(‘*’, m, n);
        return 0;
}
```

```
void stampaRettangolo (char c, int row, int col){
        int i, j;
        for (i = 0; i < row; i++) {
            for (j = 0; j < col; j++) {
                if (i == 0 || i == row-1) {
                    printf(“%c”, c);
                }
                else {
                    if (j == 0 || j == col-1) {
                        printf(“%c”, c);
                    }
                    else {
                        printf(“ ”);
                    }
                }
            }
            printf(“\n”);
        }
}
```

Avrei potuto usare la funzione stampaRettangolo anche nell’esercizio 1 semplicemente chiamando stampaRettangolo(‘*’, n, n);

Soluzione Ex. 3

```
#include <stdio.h>

void stampaQuadrato2 (char c, int x);

int main() {
        int n;
        scanf (“%d”, &n);
        stampaQuadrato2(‘*’, n);
        return 0;
}
```

```
void stampaQuadrato2 (char c, int x) {
        int i, j;
        for (i = 0; i < x; i++) {
            for (j = 0; j < x; j++) {
                if (i == 0 || i == x-1) {
                    printf(“%c”, c);
                }
                else {
                    if (j == 0 || j == x-1 || i == j ||
                        i + j == x - 1) {
                        printf(“%c”, c);
                    }
                    else {
                        printf(“ ”);
                    }
                }
            }
            printf(“\n”);
        }
}
```

Soluzione Ex. 4

```
#include <stdio.h>

int checkPerfetto (int x);

int main() {
        int n;
        scanf (“%d”, &n);
        printf(“%d\n”, checkPerfetto(n));
        return 0;
}
```

```
int checkPerfetto (int x) {
    int i;
    int count = 0;
    if (n == 0) { //0 ha inifinit divisori.
        return 0;
    }
    for (i = 1; i <= (n/2); i++) {
        if ((n%i) == 0) {
            count += i;
        }
    }
    if (count == n) {
        // controllo se e' un numero perfetto
        return 1;
    }
    else {
        return 0;
    }
}
```