Linguaggio C: le funzioni

prof. Lorenzo Porcelli

e mail: genna18@iol.it
sito: http://users.iol.it/genna18

Introduzione attraverso un esempio

Problema: Conoscendo le misure di n rettangoli, determinare l'area massima.

OB: determinare l'area più grande

RIS: areaMassima

Dati: l1, l2 di ciascun rettangolo

Limiti: l1 > 0, l2 > 0

Problema: soluzione

Supponiamo che per finire si inserisca 0 come misura di un lato.

```
Leggi l1;Leggi l2
Mentre l1 > 0 e l2 > 0
        se area(l1, l2) > areaMassima
                areaMassima = area(l1, l2)
        leggi l1; leggi l2
Scrivi areaMassima
```

Problema: soluzione

Supponiamo allora di avere a disposizione una funzione ***area*** per calcolare l'area e una funzione ***leggo*** per leggere un numero >=0.

area per operare necessita dei due lati

leggo quando interviene legge un numero positivo

Problema: algoritmo

```
l1 = leggo(); l2 = leggo()
while l1 <> 0 e l2 <> 0
        se area(l1, l2) > areaMassima
                areaMassima = area(l1, l2)
         l1 = leggo(); l2 = leggo()
Scrivo areaMassima
```

Se si ha a disposizione leggo, scrivo, area la codifica in c è semplice

Problema: codifica

```
int main() {
   float l1, l2, areaMassima = 0;

   while( (l1 = leggo()) && (l2 = leggo()) )
        if( area(l1, l2) > areaMassima )
                areaMassima = area(l1, l2);
   printf("Area massima = %f\n", areaMassima);
   return 0;
}
```

Funzioni: nuovo esempio

Sia sqrt la funzione che calcola la radice quadrata di un numero.

Problema: calcolare la radice quadrata di n numeri.

Ob: calcolare la radice quadrata

Rs: radice

Dt: numero, numero>=0

Problema: algoritmo

Supponiamo di avere a disposizione una funzione leggo che restituisce un numero positivo.

```
mentre( (x = leggo()) != 0 )
    stampa sqrt(x)
```

Esempio: codifica

```
int main() {
   double x;
   while(  (x = leggo()) != 0.0)
        printf("la radice di %lf e' %lf\n", sqrt(x) );
   return 0;
}
```

Funzione: dalla matematica

E’ una relazione tra insiemi che soddisfa la seguente proprietà:

ad ogni elemento del dominio corrisponde un unico elemento del codominio.

Sia f: X ->Y

Con y = f(x) intendiamo l’elemento y di Y che corrisponde all’elemento x di X

Come si “crea” una funzione

Definire una funzione significa descrivere ciò che compie la funzione sui dati, sul dominio, per produrre il risultato.

In C ha la seguente forma:

```
tipo nome-della-funzione (lista-parametri) {
   dichiarazioni
   istruzioni
}
```

Esempio definizione

```
double area( double b, double h) {
        return b * h;
}

double leggo( void ) {
    double x;
    do {
        printf("Inserire la misura del lato[0 per finire] ");
        scanf("%lf", &x);
    } while (x < 0);
    return x;
}
```

Esempio definizione

```
double sqrt ( double a ) {
    double y = 1, x = a;

    while(y != x) {
        x = y;
        y = 0.5 * (x + a/x);
    }
    return y;
}
```

Funzione

E’ uno strumento attraverso il quale il problema viene scomposto in parti indipendenti.

Ogni problema deve essere pensato e risolto con una sequenza di funzioni.

Un programma in C è una collezione di funzioni che interagiscono tra loro.

Esempi dalla libreria matematica

- sqrt(x) radice quadrata di *x*
- exp(x) funzione esponenziale *e*
- log(x) logaritmo naturale di *x* (in base *e*)
- log10(x) logaritmo di *x* (in base 10)
- fabs(x) valore assoluto di *x*
- ceil(x) arrotonda *x* all'intero più piccolo non minore di *x*
- floor(x) arrotonda *x* all'intero più grande non maggiore di *x*
- pow(x, y) *x* elevato alla potenza *y*
- fmod(x, y) resto di *x*/*y* in virgola mobile
- sin(x) seno trigonometrico di *x* (*x* è espressa in radianti)
- cos(x) coseno trigonometrico di *x* (*x* è espressa in radianti)
- tan(x) tangente trigonometrica di *x* (*x* è espressa in radianti)

Esempio: utilizzo

```
#include <math.h>

int main() {
        printf( "sqrt(%.1f) = %.1f\n", 900.0, sqrt( 900.0 ) );
        printf( "exp(%.1f) = %f\n", 2.0, exp( 2.0 ) );
        printf( "log(%f) = %.1f\n", 2.718282, log( 2.718282 ) );
        printf( "fabs(%.1f) = %.1f\n", 13.5, fabs( 13.5 ) );
        printf( "ceil(%.1f) = %.1f\n", 9.2, ceil( 9.2 ) );
        printf( "floor(%.1f) = %.1f\n", -9.8, floor( -9.8 ) );
        printf( "pow(%.1f, %.1f) = %.1f\n", 2.0, 7.0, pow( 2.0, 7.0 ) );
        printf( "fmod(%.3f/%.3f) = %.3f\n", 13.675, 2.333,
        printf( "sin(%.1f) = %.1f\n", 0.0, sin( 0.0 ) );

        return 0;
}
```

dichiarazione

Avviene indicando il tipo del risultato, il nome della funzione e il tipo delle variabili che servono per poter operare.

Esempio

```
float areaRettangolo( float, float);
```

areaRettangolo ha in ingresso due float e restituisce un float

In breve ….

- l'**intestazione** della definizione di funzione è costituita da tutto ciò che precede la parentesi graffa aperta
- il **corpo** della definizione di funzione è formato da ciò che si trova racchiuso tra graffe. E' un'istruzione composta che può contenere dichiarazioni.
- **l'elenco dei parametri** è tutto quello che si trova tra le due parentesi rotonde. E' una lista in cui compare il tipo del parametro e il nome
- se non compare la dichiarazione del tipo la funzione è considerata di tipo int
- una funzione che non ha parametri contiene void come lista dei parametri
- una funzione che non restituisce alcun valore è di tipo void

Osservazioni

Il problema è stato scomposto in parti per rendere la soluzione più chiara e maneggevole.

Vantaggi:

- Le singole parti sono più facili da descrivere.
- Risulta possibile riutilizzare quanto prodotto.
- La messa a punto del programma è più semplice.
- Poiché ad ogni programma occorre apportare modifiche nel corso del tempo si individua facilmente dove intervenire nel sorgente.

massimo

```
int maximum( int x, int y, int z ) {
        int max = x;
        if ( y > max ) max = y;
        if ( z > max ) max = z;

        return max;
}
```

massimo

```
#include <stdio.h>

int maximum( int, int, int ); /* prototipo della funzione */
int main() {
    int number1, number2, number3;
    scanf( “%d%d%d”, &number1, &number2, &number3 );
    printf( “Maximum is: %d\n”,
                    maximum( number1, number2, number3 ) );

    return 0;
}
```

Massimo: cosa succede?

```
int main() {
    int n1, n2, n3, n4, n5, n6, n7, n8, n9;

    scanf( “%d%d%d”, &n1, &n2, &n3 );
    scanf( “%d%d%d”, &n4, &n5, &n6 );
    scanf( “%d%d%d”, &n7, &n8, &n9 );

    printf( “Maximum is: %d\n”,
        maximum(        maximum(n1, n2, n3),
                        maximum(n4, n5, n6),
                        maximum( n7, n8, n9 )
                    )
        );
        return 0;
}
```