Laboratorio di Algoritmi e Strutture Dati

Docente: Camillo Fiorentini

9 ottobre 2007

1 Note di carratere generale

L'obiettivo è quello di scrivere delle semplici funzioni per risolvere problemi di tipo geometrico. Il passaggio di informazioni fra funzioni avviene mediante i parametri.
Si consiglia di compilare con `gcc` usando opzione `-Wall`.

1.1 Lettura di un intero da standard input

Per le operazioni di *lettura* usare la funzione `scanf()` (funzione di libreria il cui prototipo è scritto in `stdio.h`). Ad esempio, se si sono definite le variabili

```
int x, y, z;
```

con l'istruzione

```
scanf("%d", &x);
```

viene richiesto di leggere un intero da standard input e l'intero letto viene assegnato alla variabile `x`.
È possibile leggere più di un intero per volta:

```
scanf("%d%d", &x, &y);
```

legge due interi e li assegna rispettivamente alle variabili `x` e `y`.

```
scanf("%d%d%d", &x, &y, &z);
```

legge tre interi e li assegna rispettivamente alle variabili `x`, `y` e `z`.

1.2 Stampa di messaggi

Per le operazioni di *scrittura* usare la funzione `printf()` (richiede l'inclusione di `stdio.h`). Ad esempio:

```
  int n=10;
  printf("n vale %d\n",  n);
```

stampa

```
n vale 10
```

1.3 Gli operatori di confronto

Gli operatori di confronto sono: `<` (*minore*), `<=` (*minore o uguale*), `>` (*maggiore*), `>=` (*maggiore o uguale*), `==` (*uguale*), `!=` (*diverso*).

Se *Op* è uno di questi operatori, allora l'espressione E_1 *Op* E_2 vale 1 se è verificata, 0 altrimenti.

Nota

In C *non* esiste il tipo boolean (ossia, il tipo i cui elementi sono `true` e `false`), quindi le espressioni di confronto hanno tipo intero.

Ad esempio, date le definizioni

```
int x=10, y=30;
```

l'espressione `x<y` vale 1, l'espressione `x==y` vale 0.

1.4 Gli operatori NOT, AND e OR

Pur non essendo definito il tipo boolean, esistono in C degli operatori che si comportano come gli operatori booleani della logica classica: l'operatore unario `!` corrisponde al NOT, l'operatore binario `&&` all'AND, l'operatore binario `||` all'OR. Infatti:

$$!E \;=\; \begin{cases} 1 & \text{se } E = 0 \\ 0 & \text{altrimenti} \end{cases}$$

$$E_1 \texttt{ \&\& } E_2 \;=\; \begin{cases} 1 & \text{se } E_1 \neq 0 \text{ e } E_2 \neq 0 \\ 0 & \text{altrimenti} \end{cases} \qquad E_1 \texttt{ || } E_2 \;=\; \begin{cases} 1 & \text{se } E_1 \neq 0 \text{ oppure } E_2 \neq 0 \\ 0 & \text{altrimenti} \end{cases}$$

dove E, E_1 e E_2 sono espressioni che hanno un valore numerico (variabili, chiamate di funzioni, espressioni aritmetiche, ...).

Viene effettuata la *valutazione cortocircuitata (short circuit evaluation)* come in Java:

- se $E_1 = 0$, allora E_1 `&&` E_2 vale 0 e non viene calcolato il valore di E_2.
- se $E_1 \neq 0$, allora E_1 `||` E_2 vale 1 e non viene calcolato il valore di E_2.

1.5 La struttura di controllo "if-then-else"

L'istruzione

```
if(Espr)
   Istruzione1
else
   Istruzione2
```

dove `Istruzione1` e `Istruzione2` sono singole istruzioni oppure sequenze di istruzioni racchiuse fra graffe, viene eseguita nel seguente modo:

- Viene calcolato il valore E di `Espr`.
  Se $E \neq 0$, viene eseguita `Istruzione1`, altrimenti viene eseguita `Istruzione2`.

Il ramo `else` può mancare.

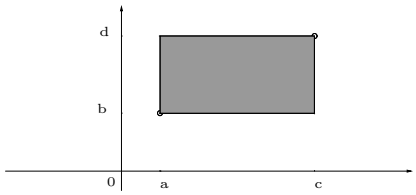
2 Il problema

Negli esercizi si fa riferimento al piano cartesiano

$$P \;=\; \{\,(x,y) \;\mid\; x,y \in \mathbf{Z}\,\}$$

dove $\mathbf{Z}$ è l'insieme dei numeri interi $\{0, 1, -1, 2, -2, \ldots\}$.
Con $R(a,b,c,d)$ denotiamo il rettangolo avente i lati paralleli agli assi cartesiani e i cui vertici in basso a sinistra e in alto a destra sono rispettivamente (a,b) e (c,d). Formalmente:

$$R(a,b,c,d) \quad = \quad \{\,(x,y) \in \mathbf{Z}^2 \;\mid\; a \le x \le c,\; b \le y \le d\,\}$$



Si chiede di scrivere alcune funzioni per risolvere alcuni problemi sui rettangoli. Ogni funzione va *documentata* con un opportuno commento e va *provata* scrivendo un programma che legge gli input necessari, chiama la funzione e stampa il risultato restituito dalla funzione, ad esempio:

```
Area rettangolo R(0,0,3,3): 9
```

Esercizo 1
Scrivere il codice di una funzione

```
int areaRett(int a, int b, int c, int d)
```

che calcola l'area del rettangolo $R(a,b,c,d)$.

Esercizo 2
Scrivere il codice di una funzione

```
int appartiene(int x, int y, int a, int b, int c, int d)
```

che restituisce 1 se $(x,y) \in R(a,b,c,d)$ (il punto (x,y) appartiene al rettangolo $R(a,b,c,d)$), 0 altrimenti.

Esercizo 3
Un rettangolo R_1 è contenuto nel rettangolo R_2 ($R_1 \subseteq R_2$) se ogni punto di R_1 appartiene a R_2. Formalmente:

$$R(a_1,b_1,c_1,d_1) \subseteq R(a_2,b_2,c_2,d_2) \quad \Longleftrightarrow \quad \forall x,y.(\,(x,y) \in R(a_1,b_1,c_1,d_1) \implies (x,y) \in R(a_2,b_2,c_2,d_2)\,)$$

Scrivere il codice di una funzione

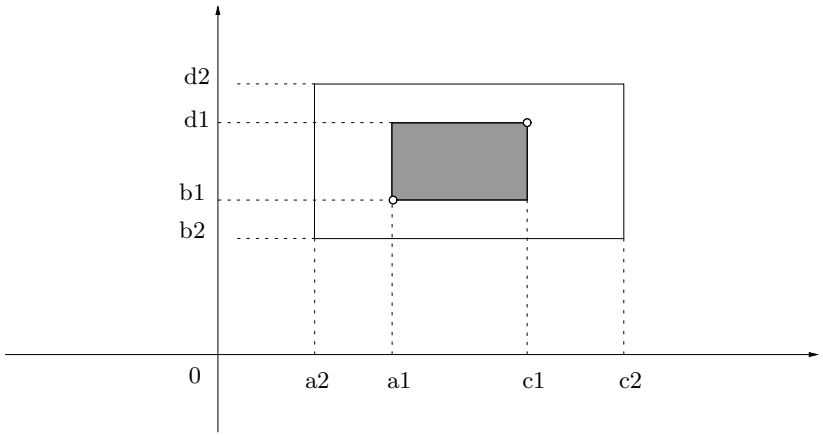
```
int contenuto(int a1, int b1, int c1, int d1, int a2, int b2, int c2, int d2)
```

che restituisce 1 se $R(a_1, b_1, c_1, d_1) \subseteq R(a_2, b_2, c_2, d_2)$, 0 altrimenti.

Suggerimento. Non è necessario controllare che *ogni* punto del primo rettangolo sia contenuto nel secondo. Infatti:

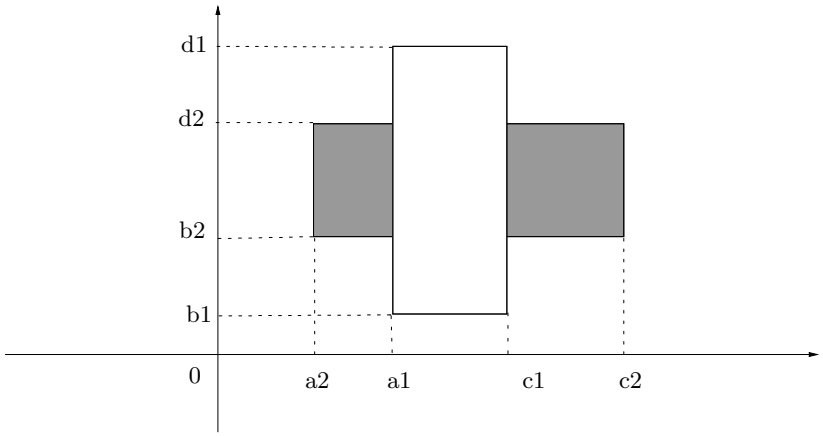
$$R(a_1, b_1, c_1, d_1) \subseteq R(a_2, b_2, c_2, d_2) \quad \Longleftrightarrow \quad ((a_1, b_1) \in R(a_2, b_2, c_2, d_2)) \ \wedge \ (c_1, d_1) \in R(a_2, b_2, c_2, d_2))$$

Cioè, $R(a_1, b_1, c_1, d_1)$ è contenuto in $R(a_2, b_2, c_2, d_2)$ se e solo se i punti (a_1, b_1) e (c_1, d_1) del primo rettangolo (evidenziati nella figura) appartengono al secondo.



Quindi l'esercizio va risolto chiamando la funzione `appartiene()` dell'Esercizio 2.

Supponiamo ora di voler sapere se due rettangoli si intersecano (cioè, se esiste almeno un punto comune ad entrambi). Diversamente dal caso precedente, non si può ragionare solo sui vertici dei rettangoli. Ad esempio, i rettangoli in figura



si intersecano, ma nessuno dei quattro vertici di un rettangolo appartiene all'altro. Si osservi tuttavia che due rettangoli si intersecano se e solo se le loro basi e le le loro altezze si "sovrapponogono". La funzione definita nel prossimo esercizio serve per determinare se due segmenti si sovrappongono.

Esercizo 4

Dati due interi s_1 e s_2 tali che $s_1 \leq s_2$, l'*intervallo* $[s_1, s_2]$ è l'insieme degli interi n tali che $s_1 \leq n \leq s_2$. Due intervalli $[s_1, s_2]$ e $[t_1, t_2]$ sono *sovrapposti* se hanno almeno un punto in comune (ossia, $[s_1, s_2] \cap [t_1, t_2] \neq \emptyset$).

Ad esempio, le seguenti coppie di intervalli si sovrappongono.

$[-10,-4]$ e $[-5,100]$, $[-1,20]$ e $[-2,10]$, $[-1,7]$ e $[7,10]$, $[3,8]$ e $[2,10]$, $[10,40]$ e $[12,15]$.

Invece gli intervalli $[-1,25]$ e $[26,30]$ non si sovrappongono.
Scrivere il codice di una funzione

```
int sovrapposti(s1, s2, t1, t2)
```

che restituisce 1 se $[s_1, s_2]$ e $[t_1, t_2]$ sono sovrapposti, 0 altrimenti.

Esercizo 5
Due rettangoli si intersecano se hanno almeno un punto in comune (l'intersezione fra i due rettangoli non è vuota). Scrivere il codice di una funzione

```
int intersezione(int a1, int b1, int c1, int d1, int a2, int b2, int c2, int d2)
```

che restituisce 1 se $R(a_1, b_1, c_1, d_1)$ e $R(a_2, b_2, c_2, d_2)$ si intersecano, 0 altrimenti.
Usare la funzione `sovrapposti()` dell'Esercizio 4 (vedere il suggerimento precedente).

Esercizo 6
Scrivere il codice di una funzione

```
int areaInt(int a1, int b1, int c1, int d1, int a2, int b2, int c2, int d2)
```

che calcola l'area dall'intersezione fra i rettangoli $R(a_1, b_1, c_1, d_1)$ e $R(a_2, b_2, c_2, d_2)$.

Suggerimento. Controllare per prima cosa che i due rettangoli si intersechino. Se i rettangoli si interscano, determinare il rettangolo $R(a, b, c, d)$ corrispondente all'intersezione fra i due rettangoli, cioè il più grande rettangolo contenuto in entrambi (per calcolare a, b, c e d è necessario definire delle funzioni `min()` e `max()` per calcolare il minimo e il massimo fra due interi). Si può ora calcolare l'area di $R(a, b, c, d)$ usando la funzione `areaRett()` dell'Esercizo 1. Se invece i due rettangoli non si intersecano, l'area dell'intersezione è 0.

2.1 Struttura del programma

```
/* file rettangoli.c */

#include <stdio.h> /* contiene prototipi di scanf() e  printf() */

/* R(a,b,c,d) denota il rettangolo formato dai punti interi  (x,y)
   tali che a<=x<=c e  b<=y<=d */


/* Calcola l'area di R(a,b,c,d) */

int areaRett(int a, int b, int c, int d){
      ...
}
```

```
...ALTRE FUNZIONI...


int main(void){
  int a1,b1,c1,d1,a2,b2,c2,d2,area;
  scanf("%d%d%d%d%d%d%d%d", &a1,&b1,&c1,&d1,&a2,&b2,&c2,&d2);
/*lettura di 8 interi */
  area =  areaRett(a1,b1,c1,d1);
  printf("Area rettangolo R(%d,%d,%d,%d): %d\n", a1,b1,c1,d1,area);
  area =  areaRett(a2,b2,c2,d2);
  printf("Area rettangolo R(%d,%d,%d,%d): %d\n", a2,b2,c2,d2,area);
  ... RISULTATI DELLE ALTRE FUNZIONI ...
  return 0;
}
```

3 Suddivisione del programma in più file

Suddividere il programma `rettangoli.c` nei seguenti file:

- Il file `rett.c` che contiene il codice delle funzioni che risolvono gli esercizi precedenti.
- Il file `rett.h` che contiene i prototipi delle funzioni in `rett.c`.
- Il file `main.c` che contiene solamente la definizione di `main()`, in cui viene chiamata una o più funzioni di `rett.c`.

  Si ricordi che è necessario includere `stdio.h` (contiene prototipi delle funzioni `printf()` e `scanf()`) e `rett.h` (prototipi delle funzioni definite in `rett.c`)

Per eseguire la compilazione generando esplicitamente i file oggetto:

```
gcc -c main.c        //  genera il file oggetto main.o
gcc -c rett.c        //  genera il file oggetto rett.o
gcc main.o rett.o    //  il linker crea l'eseguibile partendo da main.o e rett.o
```

Modificare i file sorgente ed eseguire solamente i passi necessari per rigenerare l'eseguibile.

3.1 Il comando `make`

Anziché scrivere direttamente i comandi di compilazione da effettuare, si può usare il comando `make` che esegue i comandi necessari per generare l'eseguibile in base alle regole di dipendenza in `makefile`:

```
rett: main.o rett.o
gcc  main.o rett.o -o rett

main.o: main.c rett.h
gcc  -Wall -c  main.c

rett.o: rett.c
gcc  -Wall -c rett.c
```