Compito di Programmazione

15 marzo 2010

Teoria

(1) Dato l'array char X[10][5][10], rispondere ai seguenti due punti:

(i) che tipo ha *(*X-4)+2 e che differenza c'è tra il suo valore e quello di X;

(ii) che tipo ha X[-1] e che differenza c'è tra il suo valore e quello di X;

(2) Dire se è corretto o no il seguente programma, spiegando la propria risposta:

int *f(int **p){int b=3,*x=&b; **p=b; x=*p; return x; }

main() {int y=5, b=2,*q=&b; *f(&q)=y*2;}

Programmazione iterativa:

Si tratta di fondere degli array ordinati per formare un unico array ordinato. Abbiamo un array int A[limite][20] nel quale ogni riga è ordinata in modo crescente e vogliamo con gli elementi di A riempire un altro array R di interi ad una dimensione che contenga tutti gli elementi di A in ordine crescente. La funzione che deve eseguire questo compito è la seguente:

```
int * F(int(*A)[20],int limite)
{
 int *R=new int[limite*20], *Inizio=new int[limite], r=0,k=0;
 for(int i=0; i<limite; i++) Inizio[i]=0;
 for(int i=0; i<limite*20; i++)
  {
   M(A,Inizio,limite,r,k);
   R[i]=A[r][k];
  }
 delete[] Inizio;
 return R;
}
```

R è l'array che contiene la fusione delle righe di A ed Inizio serve per ricordarci "dove siamo arrivati a prelevare" da ciascuna riga di A. Quindi ogni elemento di Inizio viene inizializzato a 0. Si chiede di realizzare la funzione iterativa M che ad ogni invocazione restituisce gli indici r e k (passati per riferimento) che individuano la posizione del minimo elemento di A tra quelli non ancora copiati in R. Ovviamente M deve sfruttare il fatto che le righe di A sono ordinate. Si chiede:

(i) scrivere la PRE e POST di M; nella PRE potete assumere che tutti gli elementi di A sono minori di MAX_INT;

(ii) scrivere il codice della funzione iterativa M che deve avere prototipo:

void M(int (*A)[20], int* Inizio, int limite, int & r, int & k); in accordo all'invocazione nel corpo di F;

(iii) per il ciclo iterativo principale di M, scrivere la condizione che deve essere vera dopo il ciclo e usare la ricetta delineata nel Capitolo 6 della dispensa per ricavare in modo automatico l'invariante del ciclo.

Programmazione ricorsiva:

Il problema è analogo al precedente, ma A è una lista tale che OGNI suo nodo punti ad un'altra lista (possibilmente vuota) "normale" ordinata in modo crescente rispetto al campo info dei suoi nodi. Con i

nodi delle liste “normali” si vuole costruire un’unica lista ordinata R. I nodi della lista A hanno tipo struct nodoP{nodo* info; nodoP*next;}; mentre le liste “normali” così come la lista R da costruire, hanno nodi del solito tipo: struct nodo{int info; nodo* next;}; La funzione ricorsiva che risolve il problema è la seguente:

```
void H(nodoP* A, nodo*& R)
{
  nodoP* x=G(A);
  if(x)
    {
      nodo*y=x->info;
      x->info=y->next;
      R=y;
      H(A,R->next);
    }
  else
     R=0;
}
```

Si chiede di realizzare la funzione ricorsiva G che ad ogni invocazione, sceglie tra le liste puntate da A quella con il primo nodo minimo e restituisce il puntatore x al nodo di A che punta a questa lista. La funzione H (che è data) toglie il primo nodo dalla lista puntata da x e lo inserisce alla fine di R. Il prototipo di G deve essere nodoP* G(nodoP*). Ovviamente alla fine di H il campo info di ogni nodo di A sarà diventato 0.

Precisamente sono richiesti i seguenti punti:

(i) scrivere PRE e POST per G;

(ii) scrivere il codice della funzione ricorsiva G con prototipo: nodoP* G(nodoP*);

(iii) dimostrare con l’induzione la correttezza di G rispetto alla PRE e POST di (i).