www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

Fondamenti di Informatica - A.A. 2016-2017
Prof. Vincenzo Caglioti
Seconda prova in itinere del **08/02/2017**

POLITECNICO MILANO 1863

Cognome	Nome	Matricola	Voto: . . . /30

Quesito:	1	2	3	4	5	Tot.
Max:	5	5	3	7	10	30
Punti:						

Istruzioni:

- per contribuire alla valutazione finale, è necessario conseguire almeno 18/30;
- non è possibile consultare libri, appunti, la calcolatrice o qualsiasi dispositivo elettronico, né comunicare;
- tempo a disposizione: 2h 00m

Stile del codice C:

- non è necessario inserire direttive `#include`;
- i commenti non sono necessari, ma potrebbero essere utili in caso di errore;
- **è consentito l'utilizzo di funzioni di libreria**.

INIZIARE LA SOLUZIONE DI OGNI ESERCIZIO SU UNA PAGINA

RESTITUIRE COMPILATO ANCHE NEL CASO IN CUI CI SI RITIRA

Quesito 1 (5 punti) Punteggio ottenuto: . . . /5

Scrivere un sottoprogramma che ricevuto in ingresso il riferimento ad un file di testo ascii crea e restituisce una lista che riporta l'insieme dei valori interi contenuti nel file, e il numero di volte che ciascuno di essi compare. Ad esempio, se il file contiene
5 0 -3 2 0 0 121 -18 5
la lista restituita sarà L →(5,2) → (0,3) → (-3, 1) → (2, 1) → (121, 1) → (-18, 1)

```
ElemLista* ContaOccorrenze(char* NomeFile)
{
  FILE* FileInput;
  ElemLista *TestaLista;
  ElemLista *NuovoElem, *UltimoElem;
  /* puntano rispettivamente ad un nuovo elemento da inserire nella lista e
   * (quando la lista non e' vuota) all'ultimo elemento della lista */
  int ValoreLetto;
  /* ultiimo numero letto dal file di input */
  ElemLista *Cursore;
  /* usato per esplorare la lista */
  boolean GiaPresente;
  /* vale true se il valore in esame e' gia' presente nella lista, false
   * altrimenti */
  int RisultatoLettura;
  /* esito di un'operazione di lettura da file */

  TestaLista = NULL;

  FileInput = fopen(NomeFile, "r");

  if ( NULL == FileInput )
  {
    printf("Errore di apertura file di ingresso!");
  }
  else
  {
    printf("\nFile di ingresso aperto correttamente.\n");

    do
    {
```

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

```
    RisultatoLettura = fscanf(FileInput, "%d", &ValoreLetto);

    if ( 1 == RisultatoLettura )
    /* se e' effettivamente stato letto un numero */
    {
      /* valutazione del numero di occorrenze del valore appena letto e
       * determinazione dell'ultimo elemento della lista */
      Cursore = TestaLista;
      GiaPresente = false;

      while (NULL != Cursore)
      {
        UltimoElem = Cursore;

        if (Cursore->Valore == ValoreLetto)
        {
          ++Cursore->NumOccorrenze;
          GiaPresente = true;
        }

        Cursore = Cursore->Prossimo;
      }

      if (false == GiaPresente)
      {
        /* creazione di un nuovo elemento e suo inserimento in coda alla
         * lista */
        NuovoElem = malloc(sizeof(ElemLista));
        NuovoElem->Valore = ValoreLetto;
        NuovoElem->NumOccorrenze = 1;
        NuovoElem->Prossimo = NULL;

        if (NULL == TestaLista)
        {
          TestaLista = NuovoElem;
        }
        else
        {
          UltimoElem->Prossimo = NuovoElem;
        }
      }
    }
  } while ( 1 == RisultatoLettura );
  /* il ciclo termina se viene raggiunta la fine del file o una riga vuota
   * (nel primo caso fscanf restituisce EOF, nel secondo restituisce 0) */

  if ( 0 != fclose(FileInput) )
  {
    printf("Errore di chiusura file di ingresso!");
  }
}

return(TestaLista);
}
```

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

Quesito2 (5 punti) Punteggio ottenuto: …/5

Una linea spezzata è una lista dinamica di punti (coordinate intere). Una linea *spezzata aperta non degenere* è tale se i suoi punti sono almeno due, e sono tutti diversi tra loro. Data una linea spezzata aperta non degenere, la sua lunghezza è la somma delle distanze euclidee tra i punti consecutivi. Date due linee spezzate aperte non degeneri A e B, A è una *scorciatoia* di B se hanno stesso punto di inizio e di fine e A è più corta di B.

Partendo dal tipo di dato punto_t riportato, si sviluppi un sottoprogramma **Scorciatoia** che, ricevute in ingresso due spezzate aperte non degeneri, restituisce 1 se la prima è una scorciatoia della seconda, 0 altrimenti.

```
typedef struct _p {
        int x, y;
        struct _p * next;} punto_t;
```

Soluzione:

```
int Scorciatoia(puntot *A, puntot *B)
{float lunA, lunB;

 if (A->x != B->x || A->y != B->y) return 0;
 lunA=0.0; lunB=0.0;
 while(A-> next != NULL)        /* scansione di A */
 {      lunA = lunA + Sqrt( (A->x - (A->next)->x))* (A->x - (A->next)->x)) +
                            (A->y - (A->next)->y))* (A->y - (A->next)->y)));
        A = A->next;
 }                              /* scnasione di B  °/
 while(B -> next != NULL)
 {      lunB = lunB + Sqrt( (B->x - (B->next)->x))* (B->x - (B->next)->x)) +
                            (B->y - (B->next)->y))* (B->y - (B->next)->y)));
        B = B->next;
 }
 if (A->x == B->x && A->y == B->y && lunA < lunB) return 1; else return 0;
}
```

Quesito 3 (3 punti) Punteggio ottenuto …/3

Che cosa stampa a schermo il seguente programma?

```
#include <stdio.h>
void Funz(char *S);

int main()
{
      char S[100] = "abcdefghilmnopq";
      Funz(S);
      return 0;
}

void Funz(char *S)
{
      if ('\0' != S[0])
      {     Funz(S+1);
            printf("%c", *S);
      }
}
```

Soluzione: il programma stampa la stringa al contrario, cioè "qponmlihgfedbca"

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

Quesito 4 (7 punti) Punteggio ottenuto .../7

Si definisca un tipo "Boolean" e un tipo atto a rappresentare una Matrice di N x N interi, con N predefinita tramite una direttiva define.
Si scriva una funzione ricorsiva che, ricevendo in ingresso una Matrice e un eventuale altro parametro utile, restituisca True se la Matrice ricevuta in ingresso è simmetrica, False in caso contrario.

Soluzione:

```
typedef enum {False, True} Boolean;
typedef int Matrice[N][N];


Boolean Simmetrica(Matrice M, int Dim)
{       int i;
        if(Dim==1) return True;
        for (i=0; i<Dim-1;i++)
                if (M[i][Dim-1] != M[Dim-1][i]) return False;
        return Simmetrica(M, Dim-1);
}
```

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari

Quesito 5 (10 punti) Punteggio ottenuto: …/10

La notazione prefissa è un modo di rappresentare espressioni algebriche senza bisogno di parentesi. Un'espressione prefissa può consistere (i) in un valore reale (rappresentato da un float) oppure (ii) in una sequenza costituita da

- un operatore, rappresentato da un carattere '+', '-', '*', oppure '/'
- un primo operando, costituito a sua volta da un'espressione prefissa
- un secondo operando, anch'esso costituito da un'espressione prefissa.

Ad esempio l'espressione * - 5.0 + 1.5 0.5 7.0 è il prodotto tra il primo operando (dato dalla differenza tra 5.0 e la somma tra 1.5 e 0.5) il cui valore pertanto è 3.0, e il secondo operando (il valore 7.0): perciò il valore dell'espressione è 21.0 .

Un'espressione prefissa è rappresentata tramite una lista dinamica di elementi:

```
typedef  struct El     {     enum{oper, val}          tipo;
                             char                     operatore;
                             float                    valore;
                             struct El                *next;  }      Elemento;
```

L'espressione dell'esempio è perciò rappresentata dalla seguente lista L:

L→(oper, '*', ...)→(oper, '-', ...)→(val, ..., 5.0)→(oper, '+', ….)→(val, ..., 1.5)→(val, ..., 0.5)→(val, …, 7.0)

Completare la funzione ricorsiva **`Calcola`** che riceve in ingresso un puntatore a un Elemento e restituisce il valore dell'espressione prefissa che comincia con l'elemento puntato. Ad esempio l'espressione che comincia con il quarto elemento della lista è la somma tra due operandi: 1.5 e 0.5. L'espressione che comincia con il terzo elemento vale 5.0. L'espressione che comincia con il secondo elemento è la differenza tra due operandi:
il primo è l'espressione che comincia con il terzo elemento, e vale 5.0; il secondo è l'espressione che comincia con il quarto elemento (la somma tra 1.5 e 0.5, che vale 2.0). L'espressione che comincia con il secondo elemento perciò vale 3.0

Soluzione:

Il punto meno ovvio è la localizzazione del secondo operando. Questo si localizza osservando che in una espressione – comunque complessa- gli elementi che precedono il secondo operando sono sempre per metà operatori e per metà valori. Partendo da questa osservazione si ricava l'algoritmo che segue:

```
float Calcola(Elemento *Punt)
{Elemento *PrimoOperando, *SecondoOperando;
 Elemento *Cursore;
 int diffValOper=0;
 if (Punt->tipo == val) return (Punt->valore);                  /* caso base, non ricorsivo */
 PrimoOperando = Punt -> next;                                  /* localizza il primo operando */
Cursore = Punt;                                                 /* localizza il secondo operando */
while (diffValOper < 0)
{  if(Cursore->tipo = val) diffValOper++ else diffValOper--;
   Cursore = Cursore -> next;}
 SecondoOperando = Cursore;
 if (Punt->operatore == '+') return (Calcola(PrimoOperando) + Calcola(SecondoOperando));
 if (Punt->operatore == '-') return (Calcola(PrimoOperando) - Calcola(SecondoOperando));
 if (Punt->operatore == '*') return (Calcola(PrimoOperando) * Calcola(SecondoOperando));
 if (Punt->operatore == '/') return (Calcola(PrimoOperando) / Calcola(SecondoOperando))
}
```

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari