ESERCIZI DI PROGRAMMAZIONE C/C++

per le classi terza

vers.0
in lavorazione

Docente SAFFI FABIO

Contenuti

Implementazione delle operazioni di base mediante main in un unico file sorgente

Struttura del file sorgente

Regole generali per costruire una applicazione basata su un unico file e senza l'utilizzo delle funzioni

La struttura del file è la seguente, non tutte le parti sono richieste nella codifica dell'algoritmo.

```
// AUTORI:
// DATA:
// TITOLO:
// COMMENTI

// #include delle librerie necessarie al programma
// #define delle costante elaborate dal preprocessore

using namespace std;
int main(){
     // DEFINIZIONI DI TIPO
     // DICHIARAZIONE DELLE COSTANTI
     // DICHIARAZIONI VARIABILI
     // MODELLAZIONE DATI
     // INIZIALIZZAZIONE VARIABILI
     // CORPO DELLA PROGRAMMA
          // INPUT
          // TRASFORMAZIONE
          // OUTPUT
     system("PAUSE");
     return 0;
}
```

Per ogni file specificare gli AUTORI, la DATA ed il TITOLO.
Nella sezione TITOLO riportare il testo dell'esercizio o una sua sintesi
Per ogni esercizio sviluppare un codice funzionante ed indicare con quali dati è stato verificato il corretto funzionamento indicare eventuali limitazione nell'uso del codice.
Tale informazioni dovranno essere inserite nella sezione COMMENTI del codice.

Organizzazione della directory di lavoro

All'interno di ogni cartella studente
- definire una cartella Informatica14-15

All'interno della cartella inserire:
-il file template con nome : TEMPLATE.cpp
-i file di esercizio con nome ESXX.cpp (XX indica il numero di esercizio a cui è riferito)

Esercizi generici

1) Dato un numero N calcolare il successivo

```
// TRASFORMAZIONE
     N=2;
     N=N+1;
```

2) Generare e visualizzare tutti i numeri compresi tra 0 e 9

```
// TRASFORMAZIONE
     for (n=0; n<=9; n++) {
          cout<<n<<endl;
     }
```

3) Generare e visualizzare tutti i numeri compresi tra 9 e 0

```
// TRASFORMAZIONE
     for (n=9; n>=0; n--) {
          cout<<n<<endl;
     }
```

Operazioni di base su una tabella di numeri interi

Nel linguaggio C/C++ una tabella viene così dichiarata:

```
#define NUM_MAX_ELEMENTI
int T[NUM_MAX_ELEMENTI];
```

4) Inizializzare una tabella T di numeri interi positivi di lunghezza L con i numeri (2,4,1,7,9)

```
// #define delle costante elaborate dal preprocessore
#define NUM_MAX_ELEMENTI

// DICHIARAZIONI VARIABILI
int T[NUM_MAX_ELEMENTI];

// INIZIALIZZAZIONE VARIABILI
     T[0]=2;
     T[1]=4;
     T[2]=1;
     T[3]=7;
     T[4]=9;
     L=5;
```

5) Data una tabella di numeri interi positivi T di lunghezza L, visualizzare tutti gli elementi.

```
// OUTPUT
     for (i=0 ; i<=L-1 ; i++) {
```

```
          cout << i <<"  "<< T[i] << endl ;
     }
```

6) Inizializzare e visualizzare una tabella T di numeri interi positivi di lunghezza L=100 con i numeri casuali positivi compresi tra 1 6

```
// #include delle librerie necessarie al programma
#include <cstdlib>
// #define delle costante elaborate dal preprocessore
#define NUM_MAX_ELEMENTI

// DICHIARAZIONI VARIABILI
int T[NUM_MAX_ELEMENTI];

// INIZIALIZZAZIONE VARIABILI
     for(i=0; i<=L-1; i++)
          T[i]= (rand()%6)+1;
```

7) Data una tabella di numeri interi positivi T di lunghezza L, inserire un elemento V in coda.

```
// TRASFORMAZIONE
     L=L+1;
     T[L-1]=V;
```

8) Data una tabella di numeri interi positivi T di lunghezza L, inserire un elemento di valore V in posizione generica P.

```
// TRASFORMAZIONE
     for( i=L-1 ; i >=P ; i--){
          T[i+1]=T[i];
     }
     L=L+1;
     T[P]=V;
```

9) Data una tabella di numeri interi positivi T di lunghezza L, cancellare l'elemento in posizione generica P.

```
// TRASFORMAZIONE
     for( i=P+1 ; i<=L-1 ; i++){
          T[i-1]=T[i];
     }
     L=L-1;
```

10) Data una tabella di numeri interi positivi T di lunghezza L, ordinare la tabella in modo crescente.

```
// TRASFORMAZIONE
     for (i=L-1 ; i>=0; i--){
          for (j=0 ; j<=i-1 ; j++) {
               if (T[j]>T[j+1]) {
                    a=T[j];
```

```
                    b=T[j+1];
                    T[j]=b;
                    T[j+1]=a;
               }
          }
     }
```

11) Data una tabella di numeri interi positivi T di lunghezza L, ricercare un dato elemento di valore V e visualizzare la relativa posizione P nella tabella; nel caso in cui l'elemento non fosse stato trovato visualizzare -1.

```
// TRASFORMAZIONE
     P=-1;
     for(i=0 ; i<=L-1; i++){
          if (T[i]==V) {
               P=i;
               break;
          }
     }
```

12) Data una tabella di numeri interi positivi T di lunghezza L, ricercare l'elemento di valore massimo MAX e la relativa posizione P.

```
// TRASFORMAZIONE
     MAX=T[0];
     P=0;
     for(i=1 ; i<=L-1; i++){
          if (T[i]>MAX) {
               MAX=T[i];
               P=i;
          }
     }
```

13) Data una tabella di numeri interi positivi T di lunghezza L, ricercare l'elemento di valore minimo MIN e la relativa posizione P.

```
// TRASFORMAZIONE
     P=0;
     MIN=T[0];
     for( i=1 ; i<=L-1; i++){
          if (T[i]<MIN) {
               MIN=T[i];
               P=i;
          }
     }
```

14) Data una tabella di numeri interi positivi T di lunghezza L, annullare la tabella cancellando tutti gli elementi.

```
// TRASFORMAZIONE
     for ( i=0 ; i<=L-1; i++) {
          T[i]=0;
```

```
     }
     L=0;
```

15) Data una tabella di numeri interi positivi T di lunghezza L, inizializzare la tabella con i valori tutti nulli.

```
// TRASFORMAZIONE
     for ( i=0 ; i<=L-1; i++) {
          T[i]=0;
     }
```

16) Data una tabella di numeri interi positivi T1 di lunghezza L1, copiare i contenuti della tabella in una seconda tabella T2.

```
// TRASFORMAZIONE
     for (i=0 ; i<=L1-1 ; i++) {
          T2[i]=T1[i] ;
     }
     L2=L1
```

17) Data una tabella di numeri interi positivi T di lunghezza L, verificare se tutti gli elementi sono uguali.

```
// TRASFORMAZIONE
     U=true;
     for( i=0 ; i<=L-2 ; i++){
          U=U && (T[i]==T[i+1]);
     }
```

18) Data una tabella di numeri interi positivi T di lunghezza L, leggere un elemento in una posizione generica P

```
// TRASFORMAZIONE
     v =T[P];
```

Operazioni di base su una tabella di caratteri

Una stringa viene vista come array di caratteri che si conclude con il valore NULL (codice ascii 0 o '\0')
Nel linguaggio C/C++ una stringa viene così dichiarata

```
#define NUM_MAX_DI_CARATTERI
char stringa[NUM_MAX_CARATTERI];
```

19) Data una stringa S, inizializzare /copiare una stringa costante o una altra stringa mediante funzione

```
// TRASFORMAZIONE
     strcpy(S,"prova");
```

20) Data una stringa di S, determinare la lunghezza L.

```
// TRASFORMAZIONE
     for ( L=0; S[L]!=0; L++);
```

21) Data una stringa S, visualizzare il contenuto (tutto su una riga).

```
// TRASFORMAZIONE
     for ( i=0; S[i]!=0; i++)
          cout <<S[i];
```

22) Date due stringhe S e S1, verificare se sono uguali

```
// TRASFORMAZIONE
     U=true;
     for (i=0; S[i]!=0; i++){
          if (S[i]==0 ){
               U=false;
               break;
          }else if (S[i]!=S1[i]){
               U=false;
               break;
          }
     }
     if (U && S1[i]!=0)
          U=false;
```

23) Date due stringhe S e S1, verificare se la stringa S precede S1 nell'ordine lessico-grafico

```
// TRASFORMAZIONE
     P=false;
     for (i=0; S[i]!=0; i++){
          if (S1[i]==0)
               break;
          else if (S[i]<S1[i]){
               P=true;
               break;
          }
     }
     if (! P&& S1[i]!=0)
          P=true;
```

24) Date due stringhe di S1 e S2, concatenarle in una unica stringa S

```
// TRASFORMAZIONE
     for ( i=0; S1[i]!=0; i++)
          S[i]=S1[i];

     for ( j=0; S2[j]!=0; j++)
          S[i+j]=S2[j];
     S[i+j]=0;
```

25) Data una stringa di S, dividerla in due sottostringhe da posizione generica P.

```
// TRASFORMAZIONE
     for ( i=0; i<=P; i++)
          S1[i]=S[i];
     S1[P+1]=0;

     for ( i=P+1; S[i]!=0; i++)
          S2[i-(P+1)]=S[i];
     S2[i-(P+1)]=0;
```

26) Date due stringhe S e S1, sostituire parte della stringa S con S1 dalla posizione P

```
// TRASFORMAZIONE
     for ( L1=0; S1[L1]!=0; L1++);

     for (i=0; i<=L1-1; i++)
          S[i+P]=S1[i];
```

27) Date due stringhe S e S1, inserire nella stringa S S1 dalla posizione P

```
// TRASFORMAZIONE
     for ( L=0; S[L]!=0; L++);
     for ( L1=0; S1[L1]!=0; L1++);

     for( i=L-1 ; i >=P ; i--)
          S[i+L1]=S[i];
     S[i+L1]=0;

     for( i=0 ; i<=L1-1; i++)
          S[i+P]=S1[i];
```

28) Data una stringa S cancellare N caratteri a partire dalla posizione P

```
// TRASFORMAZIONE
     for (i=P; S[i+N]!=0; i++)
          S[i]=S[i+N];
     S[i]=0;
```

29) Date due stringhe di S e S1, ricercare se S1 è presente in S ed in quale posizione P

```
// TRASFORMAZIONE
     P=-1;
     for ( i=0; S[i]!=0; i++){
          if (S[i]==S1[0]) {
               p=i;
               for (j=0; S1[j]!=0 ; j++){
                    if (S[i+j]!=S1[j]){
                         P=-1;
                         break;
                    }
```

```
            }
        }
        if (P!=-1) {
            break;
        }
    }
```

30) Date due stringhe S, S1 e S2 inserire nella stringa S S2 dalla posizione individuata da S1

```
// TRASFORMAZIONE
//---Trasformazione ricerca
//---trasformazione inserimento finite
```

31) Date tre stringhe S, S1 e S2, sostituire nella stringa S S1 con S2

```
// TRASFORMAZIONE
// Trasformazione di calcolo della lunghezza L1 di S1
// Trasformazione ricerca di S1 in S in posizione P
// Trasformazione di cancellazione in S di L1 caratteri in
posizione P
// Trasformazione di inserimento della stringa S2 dalla posizione
P di S
```

32) Data una stringa S convertire la lettere minuscole in lettere maiuscole

```
// TRASFORMAZIONE
    for (i=0; S[i]!=0; i++){
        if (S[i]>=97 && S[i]<=122)
            S[i]=S[i]-32;
        else
            S[i]=S[i];
    }
    S[i]=0;
```

33) Data una stringa S convertire le lettere maiuscole in minuscole

```
// TRASFORMAZIONE
    for (i=0; S[i]!=0; i++){
        if (S[i]>=65 && S[i]<=90)
            S[i]=S[i]+32;
        else
            S[i]=S[i];
    }
    S[i]=0;
```

34) Data una stringa S togliere i caratteri “SPAZIO” alla sinistra della stringa

```
// TRASFORMAZIONE
    for (n=0; S[n]==32; n++);
    for (i=0; S[i+n]!=0; i++){
        S[i]=S[i+n];
```

```
     }
     S[i]=0;
```

35) Data una stringa S togliere i caratteri “SPAZIO” alla destra della stringa

```
// TRASFORMAZIONE
     for (L=0; S[L]!=0; L++);
     for (i=L-1; i>=0; i--){
          if (S[i]!=32)break;
          S[i]=0;
     }
```

Operazioni di base su una tabella a due dimensioni di caratteri

Per questa tabella valgono le operazioni di base di base introdotte nelle precedenti sezioni ovviamente riferite alla stringa. La definizione di una tabella di stringhe è una tabella a due dimensioni la prima fa riferimento alla stringa, la seconda ai caratteri che compongono la stringa.

```
#define NUM_MAX_ELEMENTI
#define NUM_MAX_DI_CARATTERI
char TS[NUM_MAX_ELEMENTI] [NUM_MAX_CARATTERI];
```

36) Data una tabella di stringhe TS di lunghezza L ed una stringa S copiare la stringa in coda alla tabella

```
// TRASFORMAZIONE
     for (i=0; S[i]!=0; i++) TS[L][i]= S[i];
     TS[L][i]=0;
     L=L+1
```

37) Data una tabella di stringhe TS di lunghezza L copiare l’elemento della tabella in posizione P in una stringa S

```
// TRASFORMAZIONE
     for (i=0; TS[P][i]!=0; i++) S[i]= TS[P][i];
     S[i]=0;
```

Implementazione delle operazioni di base mediante funzioni

Struttura del file sorgente

Regole generali per costruire una applicazione basata su un unico file e senza l'utilizzo delle funzioni

La struttura del file è la seguente, non tutte le parti sono richieste.

```
// AUTORI:
// DATA:
// TITOLO:
// COMMENTI

// #include delle librerie necessarie al programma
// #define delle costante elaborate dal preprocessore

// prototipi delle funzioni

using namespace std;
int main(){
     // DEFINIZIONI DI TIPO
     // DICHIARAZIONE DELLE COSTANTI
     // DICHIARAZIONI VARIABILI
     // MODELLAZIONE DATI
     // INIZIALIZZAZIONE VARIABILI
     // CORPO DELLA PROGRAMMA
          // INPUT
          // TRASFORMAZIONE
          // OUTPUT
     system("PAUSE");
     return 0;
}

// funzioni
```

Riprogettare il codice sviluppato nella precedente sezione utilizzando le funzioni.
Per la progettazione delle funzioni è necessario porre attenzione a:

- i nomi da assegnare devono essere coerenti allo standard name individuato per i nomi delle variabili
- argomenti della funzioni e valori restituiti
- variabili locali a supporto dell'algoritmo implementato
- eventuali condizioni o limitazione per l’uso.

Decomposizione di un problema in sotto problemi

Implementare le operazioni di base di un insieme di numeri interi (visto come tabella)
- unione, intersezione, insieme complementare, differenza simmetrica,

Implementare le operazioni di base di un insieme di stringhe (visto come tabella a due dimensione)
- unione, intersezione, insieme complementare, differenza simmetrica,

Gestione di una coda
-lettura con cancellazione primo elemento
-inserimento in ultima posizione
-svuota tabella
-tabella vuota?

Gestione di uno stack
-lettura con cancellazione in ultima posizione
-inserimento in ultima posizione
-svuota tabella
-tabella vuota?

Implementazione delle operazioni di base mediante libreria

… da sviluppare …