ESERCIZI DI PROGRAMMAZIONE C/C++

per le classi seconde

vers.0
in lavorazione

Docente SAFFI FABIO

Contenuti

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 costanti 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;
}
```

Gli esercizi propongono la sola parte riferita alla trasformazione e raramente la parte riferita all'output.

Lo studente dovrà completare l'esercizio con le parti mancanti, compilarlo e verificarne il funzionamento indicando quali test sono stati condotti per la sua validazione. I test sono indicati nel file stesso sotto forma di commento nella sezione indicata da // commenti.

Organizzazione della directory di lavoro

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

All'interno della cartella troveremo:
-nome file del template: TEMPLATE.cpp
-nome del file dell'esercizio: ESXX.cpp

Esercizi sulle funzione di I/0 da tastiera

1. Visualizzare "Ciao" e mettere in attesa il sistema

```
// #include delle librerie necessarie al programma
#include <iostream>
// OUTPUT
     cout << "Ciao" << endl;
```

2. Visualizzare il prompt “valore:” e digitare il valore di una variabile chiamata s di tipo intero

```
// INPUT
     cout << "valore: " << endl ;
     cin >>s;
```

3. Visualizzare in colonna tre valori n1=10, n2=50, n3=30 variabili di tipo intero con distanza 20 e 30 rispettivamente

```
// #include delle librerie necessarie al programma
#include <iostream>
#include <iomanip>
// OUTPUT
     cout <<n1<<n2<<n3;
     cout << endl;

     cout <<n1<<"--"<<n2<<"--"<<n3;
     cout << endl;

     cout <<n1<<setw(20) <<n2<<setw(30)<<n3;
     cout << endl;
```

Esercizi sulla dichiarazione ed inizializzazione di variabile

4. Dichiarare i vari tipi di variabile semplice

```
// DICHIARAZIONI VARIABILI
int i;
short int si;
long int li;
float f;
double df;
char c;
bool b;

// INIZIALIZZAZIONE VARIABILI
i=0;
c='s';
b=true;
```

5. Valutare lo spazio di memoria occupato da una variabile in funzione del tipo mediante funzione sizeof (dimensione espressa in byte)

```
// OUTPUT
cout << "Dim. di int      : " << sizeof(int) << " byte" << endl;
cout << "Dim. di short int: " << sizeof(short) << " byte"<< endl;
cout << "Dim. di long int : " << sizeof(long) << " byte" << endl;
cout << "Dim. di float    : " << sizeof(float) << " byte" << endl;
cout << "Dim. di double   : " << sizeof(double) << " byte"<< endl;
cout << "Dim. di char     : " << sizeof(char) << " byte" << endl;
cout << "Dim. di bool     : " << sizeof(bool) << " byte"<<endl;
```

6. Dichiarare i vari tipi di costanti

```
// DICHIARAZIONE DELLE COSTANTI
const double PIGRECO=3.14;
const char OK='s';
const int SCONTO=20;
const int VAL1=0xAF;          //costante espressa in esadecimale
const int VAL2=077;           //costante espressa in ottale
const long MISURA=25L;
```

7. Assegnare un numero ad una variabile N

```
// TRASFORMAZIONE
     N=2;
```

Esercizi sulle funzioni di matematica

8. Dato un numero N intero positivo, calcolare il successivo

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

9. Dati due numeri interi positivi N1 e N2, calcolare il risultato della divisione intera

```
// TRASFORMAZIONE
     N=N1/N2;
```

10. Dati due numeri interi positivi N1 ed N2, calcolare il resto della divisione intera

```
// TRASFORMAZIONE
     N=N1%N2;
```

11. Visualizzare il numero più grande generabile

```
// OUTPUT
     cout << RAND_MAX << endl ;
```

12. Data un numero intero positivo N, generare un numero pseudo-casuale R intero compreso tra 0 ed N

```
// TRASFORMAZIONE
     A = rand() % N;
```

13. Dati due numeri float B ed E, calcolare la potenza P.

```
// TRASFORMAZIONE
     P=pow(B,E);
```

14. Dati un numero float positivo A calcolare la radice quadrata R

```
// TRASFORMAZIONE
     R=sqrt(A);
```

15. Dato un numero float A calcolare il numero arrotondato all'intero superiore N

```
// TRASFORMAZIONE
     N= ceil(A);
```

16. Dato un numero float A calcolare il numero arrotondato all'intero inferiore N

```
// TRASFORMAZIONE
     N= floor(A);
```

17. Dato un numero float A calcolare il numero arrotondato N

```
// TRASFORMAZIONE
     N=round(A);
```

Esercizi sulle relazioni e sui connettivi

18. Dati due numeri interi positivi N1 e N2 calcolare e visualizzare le varie relazioni matematiche

```
// OUTPUT
     cout << "dati due numeri "<<N1<<" "<<N2"<< cout;
     cout <<" N1==N2: "<< N1==N2<< endl;
     cout <<" N1>N2 : "<<N1>N2<< endl;
     cout <<" N1>=N2: "<<N1>=N2<< endl;
     cout <<" N1<N2 : "<<N1<N2<< endl;
     cout <<" N1<=N2: "<<N1<=N2<< endl;
     cout <<" N1!=N2: "<<N1!=N2<< endl;
```

Due diverse relazioni possono essere collegate fra loro a formare una nuova relazione più complessa, mediante operatori logici detti connettivi logici :
|| (connettivo OR), && (connettivo AND) ,! (NOT)

19. Dati N1=7, N2=6, N3=11, calcolare il valore delle seguenti espressioni logiche:

```
// TRASFORMAZIONE
```

```
     B1= (N1<N2) || (N3<N1) ;
     B2= !(N1==N3) && ((N1==N2) || (N2<N3));
     B3= (N1>N2) && ((N3<N1) || (N2==N3)));
```

20. Dati N1=1,N2=6, N3=6, calcolare il valore delle seguenti espressioni logiche:

```
// TRASFORMAZIONE
     B1=(N1>N3) || (N3<=N2);
     B2=(N1==N3) || ((N1<N2) && (N2<=N3));
     B3=!(N1>N2) && ((N1>N3) || (N3>=N2));
```

21. Dati N1=12, N2=9, N3=10, calcolare il valore delle seguenti espressioni logiche:

```
// TRASFORMAZIONE
     B1= (N1>N2) || (c>N1);
     B2= !(N1==N3) || ((N1<N2) && (N1<N3));
     B3= (N1>N2) && ((N3>N1) || !(N3>N2)) ;
```

Esercizi sulle regole di conversione

22. Regola di conversione (implicità) da float ad intero

```
// DICHIARAZIONI VARIABILI
     int a;
     float b;
// INIZIALIZZAZIONE VARIABILI
     b=3.56;
// TRASFORMAZIONE
     a=b;
// OUTPUT
     cout << a << "\n";
```

23. Conversione da float ad intero

```
// DICHIARAZIONI VARIABILI
     int a;
     float b;
// INIZIALIZZAZIONE VARIABILI
     b=3.56;
// TRASFORMAZIONE
     a=(int)b;
// OUTPUT
     cout << a << endl;
```

24. Regola di conversione (implicita) da intero a float

```
// DICHIARAZIONI VARIABILI
     int a;
     float b;
// INIZIALIZZAZIONE VARIABILI
     a=5;
// TRASFORMAZIONE
```

```
     b=a;
// OUTPUT
     cout << b << endl;
```

Esercizi sulla struttura di controllo di selezione

25. Dati tre numeri interi positivi N1 N2 N3, calcolare e visualizzare il numero max

```
// TRASFORMAZIONE
     Nmax=N1;
     if (N2>Nmax) Nmax=N2;
     if (N3>Nmax) Nmax=N3;
     cout<<Nmax<< endl;
```

26. Dato un numero intero positivo N visualizzare i primi tre valori in un testo UNO, DUE, TRE

```
// OUTPUT
     if (N==1) cout << "UNO" << endl;
     else if (N==2) cout << "DUE" << endl;
     else if (N==3) cout << "TRE" << endl;
     else cout << "VALORE SUPERIORE A TRE" << endl;
```

Esercizi sul ciclo iterattivo FOR

27. Visualizzare tutti i numeri da 0 a 9 compresi

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

28. Visualizzare tutti i numeri da 9 a 0 compresi

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

29. Dato N un numero intero positivo, visualizzare in ordine crescente i numeri dispari minori o uguali a N

```
// TRASFORMAZIONE
     for (i=1; i<=N; i=i+2){
          cout<< i<< endl;
     }
```

30. Calcolare la somma S dei primi N numeri

```
// TRASFORMAZIONE
     S=0;
     for(i=1;i<=N;i=i+1){
          S=S+i;
```

```
}
```

31. Calcolare il prodotto P dei primi N numeri

```
// TRASFORMAZIONE
     P=1;
     for(i=1;i<=N;i=i+1){
          P=P*i;
     }
```

32. Calcolare la potenza P in base B ed esponente E

```
// TRASFORMAZIONE
     P=1;
     for(i=1;i<=E;i=i+1){
          P=P*B;
     }
     cout<<P<<endl;
```

33. Dati due numeri interi positivi N1 N2, calcolare il prodotto P di N1*N2 mediante la ripetizione dell'operazione somma

```
// TRASFORMAZIONE
     P=0;
     for(i=1;i<=N2;i=i+1){
          P=P+N1;
     }
```

34. Dati due numeri interi positivi N1 N2 con N1>N2, calcolare il quoziente ed il resto della divisione N1/N2 mediante la ripetizione dell'operazione sottrazioni

```
// TRASFORMAZIONE
     Q=0;
     for(R=N1; R>=N2; R=R-N2)
     Q=Q+1;
```

35. Visualizzare in colonna a gruppi di tre con distanza 10 i primi 18 numeri

```
// OUTPUT
     for (i=1; i<=18; i++){
          if ((i-1)%3>0) {
               cout <<i<< setw(10);
          }else {
               cout << endl<< i << setw(10);
     }
```

36. Visualizzare la tabella pitagorica

```
// TRASFORMAZIONE
     for(n=1; n<=10; n++) {
          for(i=1; i<=10; i++) {
```

```
            cout <<" ";
            cout <<n*i;
        }
        cout <<endl;
    }
```