**I.I.S. “B. Pascal” Reggio Emilia**

**VERIFICA DI INFORMATICA**

**Classe \_\_\_\_\_\_\_\_\_\_\_\_\_Nome\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Data: 24 Ottobre 2018**

1. Dati i seguenti metodi per la classe vector illustrane il significato (p..……/10)

**size() restituisce la dimensione attuale del vettore**

**push\_back(**) **aggiunge un elemento in fondo**

**pop\_back()** **elimina ultimo elemento**

**begin()** **ritorna l’iteratore al primo elemento**

**clear() svuota il vettore**

1. Data la seguente lista:

list<char> l;

e supponendo che contenga i seguenti elementi: a c b d h g i

scrivere nella colonna di destra il contenuto della lista l dopo l’esecuzione del codice indicato nella colonna a sinistra: (p………/9)

|  |  |
| --- | --- |
| CODICE | CONTENUTO DI l |
| list<char>::iterator i ;  l.push\_back(e);  for (i = l.begin(); i != l.end(); iter++)  cout << \*i <<endl; | **a c b d h g i e** |
| list<char>::iterator i ;  sort( l.begin(),l.end() );  for (i = l.begin(); i != l.end(); iter++)  cout << \*i <<endl; | **a b c d g h i** |
| list<char>::iterator i ;  reverse( l.begin(),l.end() );  for (i = l.begin(); i != l.end(); iter++)  cout << \*i <<endl; | **i h g d c b a** |

1. Che cosa viene restituito dal metodo find() del contenitore map **m: m.find(T1)** ? (p……………/2)

**Un iteratore a (T1, valore) (cioè corrispondente alla posizione trovata) ; se non lo trova restituisce l’iteratore m.end()**

1. Completa la seguente tabella specificando con una lettera A o con una lettera M se le funzioni elencate sono algoritmi (A) o metodi (M) per i contenitori indicati sulle colonne (p……./9)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **vector** | **list** | **map** |
| **count()** | A | A | M |
| **for\_each** | A | A | A |
| **unique()** | A | M | A |

1. Dato un vettore di numeri di dimensione terminante conla richiesta all’utente “Altro numero (0=FINE)?”, memorizzare per ciascun numero in un altro vettore il doppio del numero se il numero è pari, il numero stesso se è dispari. Visualizzare il secondo vettore. Utilizzare la libreria STL. (p………/50)

1//ese.verifica….

1#include <iostream>

2#include <vector>

2#include <algorithm>

1using namespace std;

3vector<int> v1, v2;

10void caricaVettore()

{

int d;

**// inserimento componenti**

cout<< "Numero (0=fine): ";

cin >> d;

**//aggiunge i componenti**

while (d != 0) {

v1.push\_back(d);

cout << "Altro numero (0=fine): ";

cin >> d;

}

}

10void creaVettore( int num)

{

if(num%2==0) v2.push\_back(2\*num);

else v2.push\_back(num);

}

10void visualizza()

{

vector<int>::iterator i ;

cout<<"il vettore creato \x8a: "<<endl;

for (i= v2.begin(); i!= v2.end(); i++)

cout << \*i <<endl;

}

10 int main()

{

caricaVettore();

for\_each(v1.begin(),v1.end(),creaVettore);

visualizza();

system("pause");

return 0;

}

1. Acquisire da tastiera un elenco di parole, memorizzandole in una lista, finchè l’utente segnala la fine dell’inserimento con l’”\*”.

Dichiarare le librerie necessarie per l’utilizzo della struttura lista, dichiarare la lista e svolgere SOLO inserimento in testa delle parole nelle due modalità. (p……./24)

**Primo modo**

1 #include <iostream>

#include<string>

3 struct Nodo {

string parola;

Nodo\* next;

};

**8 Nodo\* InserisciInTesta(Nodo\* &testa, Nodo\* nuovo)**

{

if (testa==NULL) testa = nuovo;

else

{

nuovo->next=testa;

testa=nuovo;

}

return(testa);

}

**Secondo modo**

1,5 #include<iostream>

#include<list>

#include<string>

2.5 list <string> l;

**8 InserisciInTesta()**

**{**

string d;

// inserimento componenti

cout<< "Parola (‘\*’=fine): ";

cin >> d;

//aggiunge i componenti della lista

while (d != ‘\*’) {

l.push\_front(d);

cout << "Altro numero (‘\*’=fine): ";

cin >> d;

}

1. Che cosa scrive in output il seguente programma? (p………/6)

//ese7.cpp

#include <iostream>

#include <map>

#include <string>

#include <iterator>

using namespace std;

int main()

{

map<string, int> mapOfWords;

// Inserting data in std::map

mapOfWords.insert(make\_pair("earth", 1));

mapOfWords.insert(make\_pair("moon", 2));

mapOfWords["sun"] = 3;

// Will replace the value of already added key i.e. earth

mapOfWords["earth"] = 4;

// Iterate through all elements in std::map

map<std::string, int>::iterator it = mapOfWords.begin();

while(it != mapOfWords.end())

{

cout<<it->first<<" :: "<<it->second<<endl;

it++;

}

// Check if insertion is successful or not

if(mapOfWords.insert(make\_pair("earth", 1)).second == false)

{

cout<<"Element with key 'earth' not inserted because already existed"<<endl;

}

// Searching element in std::map by key.

if(mapOfWords.find("sun") != mapOfWords.end())

cout<<"word 'sun' found"<<endl;

if(mapOfWords.find("mars") == mapOfWords.end())

cout<<"word 'mars' not found"<<endl;

return 0;

}

OUTPUT:

**earth :: 4  
moon :: 2  
sun :: 3  
Element with key ‘earth’ not inserted because already existed  
word ‘sun’ found  
word ‘mars’ not found**