Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Soluzione:

*B012 UNO*

*Bc B012 Bc b C012 DUE*

*B012 B012 Bc b C012 TRE*

*Bc Bc QUATTRO*

*ebd CINQUE*

Immagine che contiene testo, schermata, design

Descrizione generata automaticamente

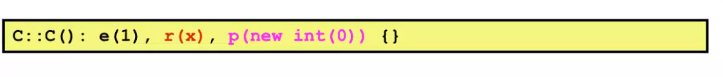


Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Soluzione:

*UNO*

*N2 N2 N2 N2 DUE*

*pluto~N paperino~N topolino~N pippo~N ~C() TRE*

*Immagine che contiene testo, schermata, software

Descrizione generata automaticamente*

Soluzione

E::h(c); // NON COMPILA -> h chiede const &;

        // non c'è nessun convertitore di tipo da classe C ad E

c.g(); // NON COMPILA -> La classe C non c'è nessun metodo g

E::h(d); // COMPILA -> Nonostante il convertitore di tipo ci sia, la conversione viene fatta

        // in maniera implicita

e.i(); // COMPILA -> This dereferenziato viene convertito da E in C (r.15)

C::f(d); // NON COMPILA -> Viene invocata D ma al suo interno non c'è un covertitore da D a C (r.9)

C::f(e); // COMPILA -> Per via del convertitore di tipo (r.5 into r.14)

d.i(); // NON COMPILA -> La classe D non ha il metodo i

E e1(c); // COMPILA -> Oggetto tipo C convertito in D (r.9) poi convertito in E (r.14)

D d1(c); // COMPILA -> Oggetto tipo C convertito in D (r.9)

C c1(e); // COMPILA -> Non c'è un costruttore abile a fare conversione C>E, però lo fa operator

C c2(d); // NON COMPILA -> Perchè in classe C nessun convertitore D>C

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

 int main() {

   cout << "\*\*1\n"; // 9 C01

   C c(7); cout << "\*\*2\n"; // 7 C01

   D d; cout << "\*\*3\n"; // 6 C01 8 C01 D0

   c = \*(d.getUno()); cout << "\*\*4\n"; // ""

   c = d.z1.k; cout << "\*\*5\n"; // 6 C01

   E e1(d); cout << "\*\*6\n"; // 6 C01 6 C01 Dc 6 C01 8 C01 D0 E(D)

   E e2 = e1; cout << "\*\*7\n"; // 6 C01 8 C01 D0 Ec

   e2 = e1; cout << "\*\*8"; // E=

}

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

 int main() {

   Lista x1; cout << " \*\*1\n"; // \*\*1

   //non stampa nulla

   Lista x2(5); cout << " \*\*2\n"; // ~S ~S ~N ~S ~S ~S \*\*2

   Lista\* p = new Lista(3); cout << " \*\*3\n"; //  ~S ~S ~N ~S ~S ~S \*\*3

   delete p; cout << " \*\*4\n"; // ~N ~N ~S ~S ~S \*\*4

   Lista x3(0); cout << " \*\*5\n"; // ~S \*\*5

   x2.remove(); cout << " \*\*6\n"; // ~S ~S ~N ~S ~S ~S \*\*6

}

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteImmagine che contiene testo, linea, Carattere, schermata

Descrizione generata automaticamente

*Soluzione*

2 5 2 0x7ffee4ff1194 0x7ffee4ff1194

Immagine che contiene testo, schermata, Carattere, schermo

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

*Soluzione*

*0 1 2 3 UNO*

*0 1 3 6 DUE*

*Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente*

#include <iostream>

#include <string>

using namespace std;

class Data {

    private:

        int giorno, mese, anno;

        string giorno\_settimana;

    public:

        Data(int g, int m, int a, string gs) {

            giorno = g;

            mese = m;

            anno = a;

            giorno\_settimana = gs;

        }

        Data(int g, int m, int a) {

            giorno = g;

            mese = m;

            anno = a;

            giorno\_settimana = "";

        }

        Data() {

            giorno = 0;

            mese = 0;

            anno = 0;

            giorno\_settimana = "";

        }

        int getGiorno() const {

            return giorno;

        }

        int getMese() const {

            return mese;

        }

        int getAnno() const {

            return anno;

        }

        string getGiornoSettimana() const{

            return giorno\_settimana;

        }

        void setGiorno(int g) {

            giorno = g;

        }

        void setMese(int m) {

            mese = m;

        }

        void setAnno(int a) {

            anno = a;

        }

        void setGiornoSettimana(string gs) {

            giorno\_settimana = gs;

        }

        void aggiungi\_uno() {

            if (giorno == 31 && mese == 12) {

                giorno = 1;

                mese = 1;

                anno++;

            } else if (giorno == 31 && (mese == 1 || mese == 3 || mese == 5 || mese == 7 || mese == 8 || mese == 10)) {

                giorno = 1;

                mese++;

            } else if (giorno == 30 && (mese == 4 || mese == 6 || mese == 9 || mese == 11)) {

                giorno = 1;

                mese++;

            } else if (giorno == 28 && mese == 2) {

                giorno = 1;

                mese++;

            } else {

                giorno++;

            }

        }

        bool operator==(Data d) {

            if (giorno == d.getGiorno() && mese == d.getMese() && anno == d.getAnno()) {

                return true;

            } else {

                return false;

            }

        }

        bool operator<(Data d) {

            if (anno < d.getAnno()) {

                return true;

            } else if (anno == d.getAnno() && mese < d.getMese()) {

                return true;

            } else if (anno == d.getAnno() && mese == d.getMese() && giorno < d.getGiorno()) {

                return true;

            } else {

                return false;

            }

        }

        friend ostream& operator<<(ostream& os, const Data& data) {

            os << data.getGiornoSettimana() << " " << data.getGiorno() << "/" << data.getMese() << "/" << data.getAnno();

            return os;

        }

};

// Esemplificare l'uso della classe e di tutti i suoi metodi tramite un esempio di main()

int main() {

    // Creiamo una data con giorno della settimana

    Data data1(21, 10, 2002, "lun");

    // Creiamo una data senza giorno della settimana

    Data data2(31, 1, 2002);

    // Visualizziamo le date utilizzando l'overloading dell'operatore di output

    cout << "Data 1: " << data1 << endl;

    cout << "Data 2: " << data2 << endl;

    // Verifichiamo l'uguaglianza tra due date

    if (data1 == data2) {

        cout << "Le date sono uguali." << endl;

    } else {

        cout << "Le date non sono uguali." << endl;

    }

    // Verifichiamo l'operatore relazionale "<" ignorando il giorno della settimana

    if (data1 < data2) {

        cout << "Data 1 è precedente a Data 2." << endl;

    } else {

        cout << "Data 2 è precedente a Data 1." << endl;

    }

    // Aggiungiamo un giorno alla data

    data1.aggiungi\_uno();

    cout << "Dopo l'aggiunta di un giorno: " << data1 << endl;

    return 0;

}