adania pobieramy stąd: <http://info.wsisiz.edu.pl/~olbratow/ns_po1/download.html>

Rozwiązania uploadujemy tutaj: <http://info.wsisiz.edu.pl/~olbratow/ns_po1/upload.html>

Kolosy

ktoś error robi?

**Zadanie ERROR**

**Napisz klasę Error pamiętającą opis i kod błędu. Klasa powinna być przystosowana do użycia w przykładowych programach poniżej. Klasa korzysta tylko z pliku nagłówkowego string.**

**Przykładowy program 1**

**int main() {**

**Error error;**

**std::string description = error.description();**

**int code = error.code();**

**std::cout << code << std::endl; }**

**Wykonanie przykładowego programu 1**

**Out: -1**

**Przykładowy program 2**

**int main() {**

**Error error1("invalid argument", 2), error2;**

**(error2 = error1) = error1;**

**Error error3(error2);**

**std::cout << error3.description() << " " << error3.code() << std::endl; }**

**Wykonanie przykładowego programu 2**

**Out: invalid argument 2**

**Przykładowy program 3**

**int main() {**

**std::vector<Error> errors;**

**std::string description;**

**int code;**

**while (std::cin >> description >> code) {**

**errors.push\_back(Error(description, code)); }**

**std::sort(errors.begin(), errors.end());**

**for (auto iterator = errors.cbegin(); iterator < errors.cend(); ++iterator) {**

**std::cout << iterator->description() << " " << iterator->code() << std::endl; }}**

**Opis przykładowego programu 3**

**\begin{flushleft} Ten program sortuje błędy rosnąco według ich kodów. Błędy o jednakowych kodach sortuje rosnąco w alfabetycznej kolejności opisów. \end{flushleft}**

**#include <iostream>**

**#include <string>**

**#include <vector>**

**#include <algorithm>**

**class Error**

**{**

**private:**

**int kod;**

**std::string desc;**

**public:**

**Error()**

**{**

**kod = -1;**

**desc = "";**

**}**

**Error(const Error &e2)**

**{**

**\*this = e2;**

**}**

**Error& operator =(const Error& e2)**

**{**

**kod = e2.kod;**

**desc = e2.desc;**

**return \*this;**

**}**

**Error(std::string str, int num)**

**{**

**desc = str;**

**kod = num;**

**}**

**std::string description() const**

**{**

**return this->desc;**

**}**

**int code() const**

**{**

**return this->kod;**

**}**

**bool operator<(Error const &e) const**

**{**

**return (kod < e.kod) ||**

**((kod == e.kod) && (desc < e.desc));**

**}**

**};**

**zadanie FLIGHT**

**Napisz klasę Flight pamiętającą nazwę miasta docelowego i godzinę odlotu zapisaną jako liczba całkowita, przykładowo 1623 to szesnasta dwadzieścia trzy. Klasa powinna być przystosowana do użycia w przykładowych programach poniżej. Klasa korzysta tylko z pliku nagłówkowego string.**

**Przykładowy program 1**

**int main() {**

**Flight flight;**

**std::string destination = flight.destination();**

**int departure = flight.departure();**

**std::cout << departure << std::endl; }**

**Wykonanie przykładowego programu 1**

**Out: 1200**

**Przykładowy program 2**

**int main() {**

**Flight flight1("Warsaw", 1615), flight2;**

**(flight2 = flight1) = flight1;**

**Flight flight3(flight2);**

**std::cout << flight3.destination() << " " << flight3.departure() << std::endl; }**

**Wykonanie przykładowego programu 2**

**Out: Warsaw 1615**

**Przykładowy program 3**

**int main() {**

**std::vector<Flight> flights;**

**std::string destination;**

**int departure;**

**while (std::cin >> destination >> departure) {**

**flights.push\_back(Flight(destination, departure)); }**

**std::sort(flights.begin(), flights.end());**

**for (auto iterator = flights.cbegin(); iterator < flights.cend(); ++iterator) {**

**std::cout << iterator->destination() << " " << iterator->departure() << std::endl; }}**

**Opis przykładowego programu 3**

**\begin{flushleft} Ten program sortuje loty rosnąco w alfabetycznej kolejności miast docelowych. Loty do tego samego miasta sortuje malejąco według godzin odlotu. \end{flushleft}**

**zadanie ADDRESS**

**Napisz klasę Address pamiętającą nazwę ulicy i numer domu. Klasa powinna być przystosowana do użycia w przykładowych programach poniżej. Klasa korzysta tylko z pliku nagłówkowego string.**

**Przykładowy program 1**

**int main() {**

**Address address;**

**std::string street = address.street();**

**int number = address.number();**

**std::cout << number << std::endl; }**

**Wykonanie przykładowego programu 1**

**Out: 1**

**Przykładowy program 2**

**int main() {**

**Address address1("Newelska", 6), address2;**

**(address2 = address1) = address1;**

**Address address3(address2);**

**std::cout << address3.street() << " " << address3.number() << std::endl; }**

**Wykonanie przykładowego programu 2**

**Out: Newelska 6**

**Przykładowy program 3**

**int main() {**

**std::vector<Address> addresses;**

**std::string street;**

**int number;**

**while (std::cin >> street >> number) {**

**addresses.push\_back(Address(street, number)); }**

**std::sort(addresses.begin(), addresses.end());**

**for (auto iterator = addresses.cbegin(); iterator < addresses.cend(); ++iterator) {**

**std::cout << iterator->street() << " " << iterator->number() << std::endl; }}**

**Opis przykładowego programu 3**

**\begin{flushleft} Ten program sortuje adresy rosnąco w alfabetycznej kolejności nazw ulic. Adresy przy tej samej ulicy sortuje według rosnących numerów domu. \end{flushleft}**

**zadanie CAR**

**Napisz klasę Car pamiętającą nazwę modelu samochodu oraz liczbę litrów paliwa spalanych na sto kilometrów. Klasa powinna być przystosowana do użycia w przykładowych programach poniżej. Klasa korzysta tylko z pliku nagłówkowego string.**

**Przykładowy program 1**

**int main() {**

**Car car;**

**std::string model = car.model();**

**double burnup = car.burnup();**

**std::cout << burnup << std::endl; }**

**Wykonanie przykładowego programu 1**

**Out: 5**

**Przykładowy program 2**

**int main() {**

**Car car1("fso 1500", 10.5), car2;**

**(car2 = car1) = car1;**

**Car car3(car2);**

**std::cout << car3.model() << " " << car3.burnup() << std::endl; }**

**Wykonanie przykładowego programu 2**

**Out: fso 1500 10.5**

**Przykładowy program 3**

**int main() {**

**std::vector<Car> cars;**

**std::string model;**

**double burnup;**

**while (std::cin >> model >> burnup) {**

**cars.push\_back(Car(model, burnup)); }**

**std::sort(cars.begin(), cars.end());**

**for (auto iterator = cars.cbegin(); iterator < cars.cend(); ++iterator) {**

**std::cout << iterator->model() << " " << iterator->burnup() << std::endl; }}**

**Opis przykładowego programu 3**

**\begin{flushleft} Ten program sortuje samochody według malejącego zużycia paliwa. Samochody o jednakowych zużyciach paliwa sortuje rosnąco w alfabetycznej kolejności modeli. \end{flushleft}**

class Car {

public:

std::string name = "";

double fuel = 5;

Car() {}

Car(std::string model, double burnUp) {

name = model;

fuel = burnUp;

}

double burnup() const {

return fuel;

}

std::string model() const {

return name;

}

bool testModelName(std::string modelName) const {

return name < modelName;

}

bool operator<(Car const &car) const

{

return car.fuel < fuel

|| (testModelName(car.name) && fuel == car.fuel);

}

};