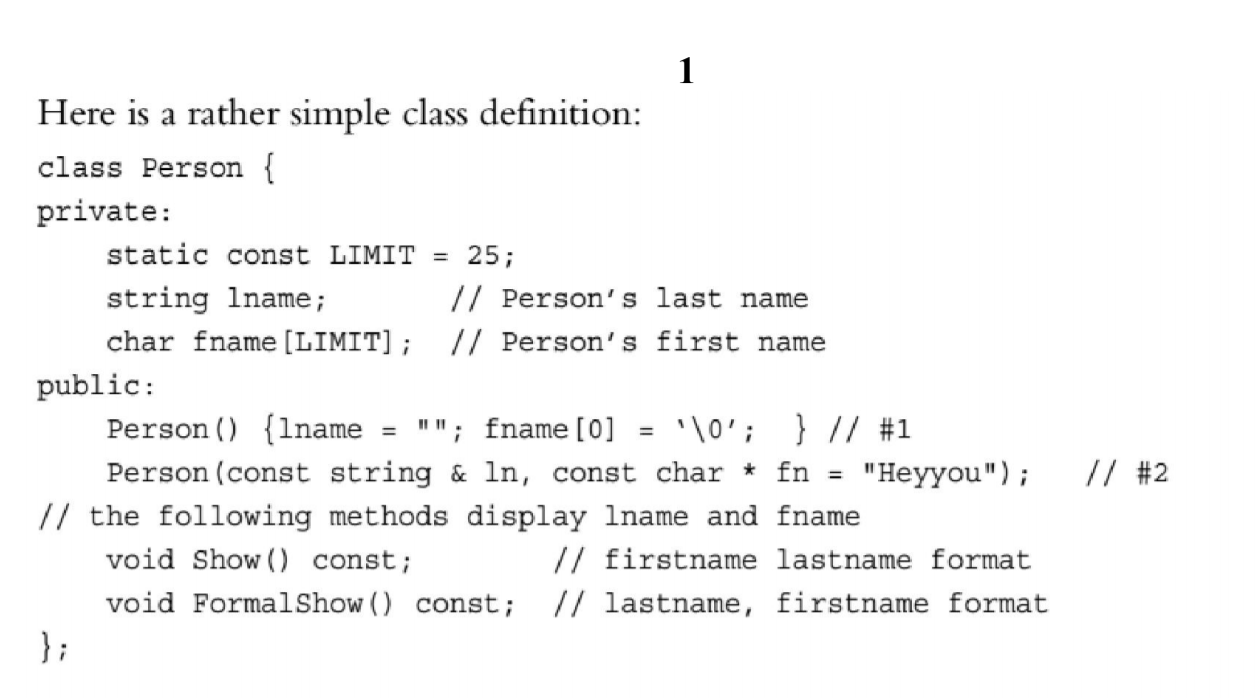
Oleynik Vladislav

Exercise 15



#include <iostream>

#include <string>

using namespace std;

class Person

{

private:

static const int LIMIT = 25;

string lname;

char fname[LIMIT];

public:

Person()

{

lname = "";

fname[0] = '\0';

}

Person(const std::string &ln, const char \*fn = "Test");

void Show() const;

void FormalShow() const;

};

Person::Person(const std::string &ln, const char \*fn)

{

lname = ln;

strcpy(fname, fn);

}

void Person::Show() const

{

cout << this->fname << " " << this->lname << endl;

}

void Person::FormalShow() const

{

cout << this->lname << " " << this->fname << endl;

}

int main()

{

Person person\_1;

person\_1.Show();

person\_1.FormalShow();

string name = "Vlad";

char \*surname = (char \*)"Vkus";

Person person\_2(name);

person\_2.Show();

person\_2.FormalShow();

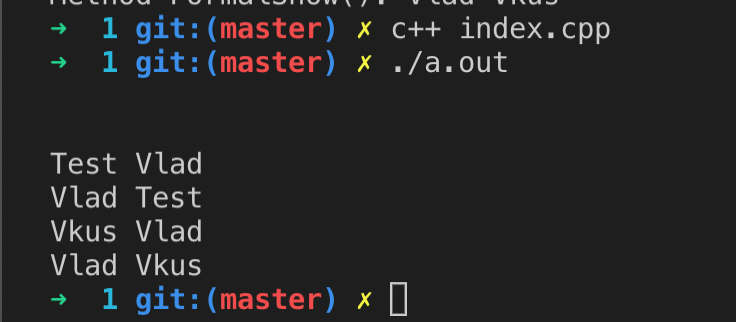
Person person\_3(name, surname);

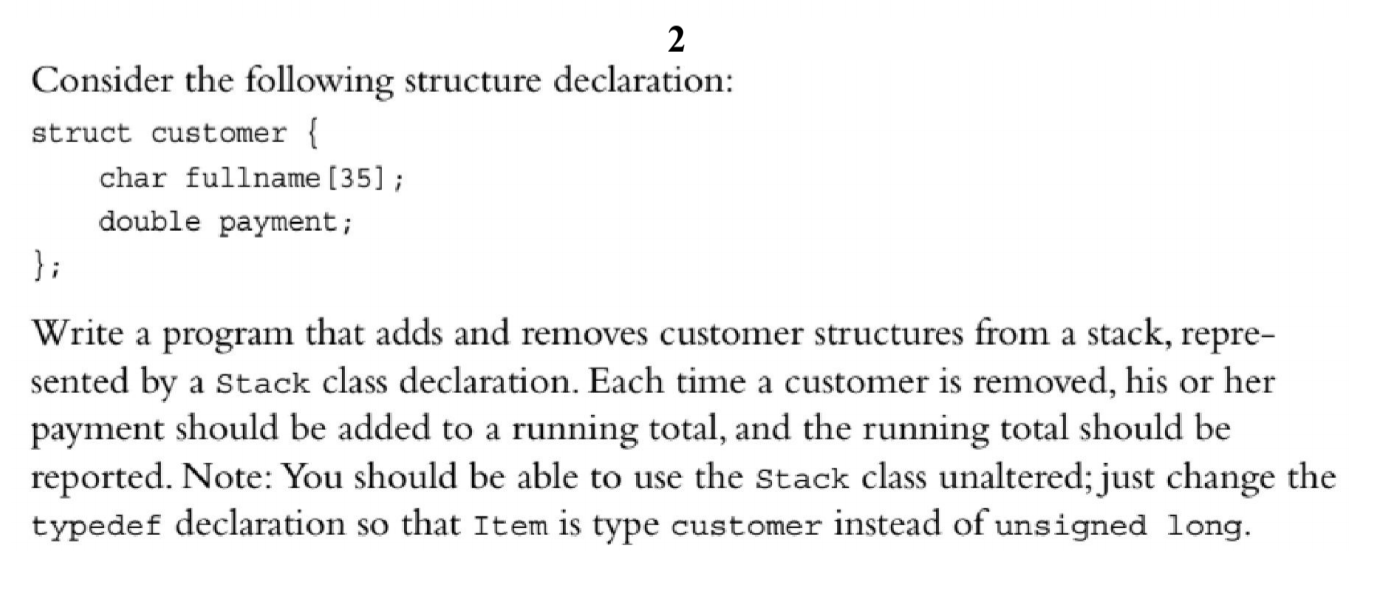
person\_3.Show();

person\_3.FormalShow();

}

Output:





#include <iostream>

#include <string>

#include <cstring>

struct customer

{

char fullname[35];

double payment;

};

typedef customer Item;

class Stack

{

private:

enum

{

MAX = 10

};

Item items[MAX];

int top;

public:

Stack();

bool isempty() const;

bool isfull() const;

bool push(const Item &item);

bool pop(Item &item);

};

Stack::Stack()

{

top = 0;

}

bool Stack::isempty() const

{

return top == 0;

}

bool Stack::isfull() const

{

return top == MAX;

}

bool Stack::push(const Item &item)

{

if (top < MAX)

{

items[top++] = item;

return true;

}

else

return false;

}

bool Stack::pop(Item &item)

{

if (top > 0)

{

item = items[--top];

return true;

}

else

return false;

}

int main()

{

using std::cin;

using std::cout;

using std::endl;

using std::string;

Stack myStack;

int selection = 0;

double total = 0;

cout << "Enter your selection" << endl;

cout << "1 - Add customer" << endl;

cout << "2 - Remove customer" << endl;

cout << "Any other character - quit" << endl;

while (cin >> selection && (selection == 1 || selection == 2))

{

cin.get();

if (selection == 1)

{

if (myStack.isfull())

{

cout << "Cannot add customer. Stack is full." << endl;

}

else

{

customer newCustomer;

cout << "Enter customer name: ";

cin.getline(newCustomer.fullname, 35);

cout << "Enter payment: ";

(cin >> newCustomer.payment).get();

myStack.push(newCustomer);

cout << "Customer " << newCustomer.fullname << " added with a payment of " << newCustomer.payment << endl;

}

}

else

{

if (myStack.isempty())

{

cout << "Cannot remove customer. Stack is empty." << endl;

}

else

{

customer aCustomer;

myStack.pop(aCustomer);

cout << "Customer " << aCustomer.fullname << " removed." << endl;

total += aCustomer.payment;

cout << "Running Total: " << total << endl;

}

}

cout << "Make another selection: ";

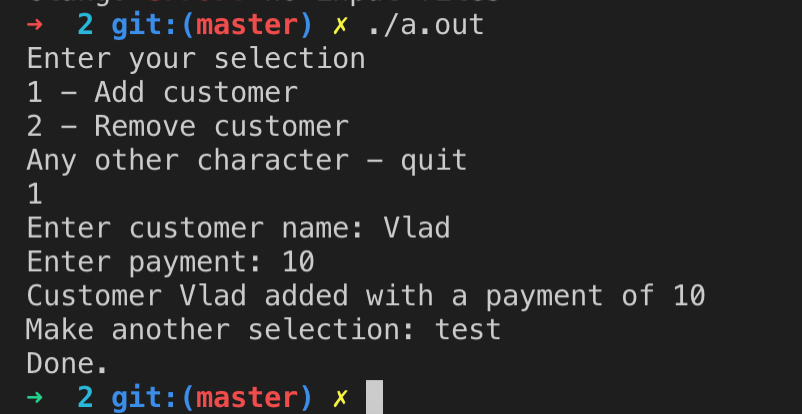
}

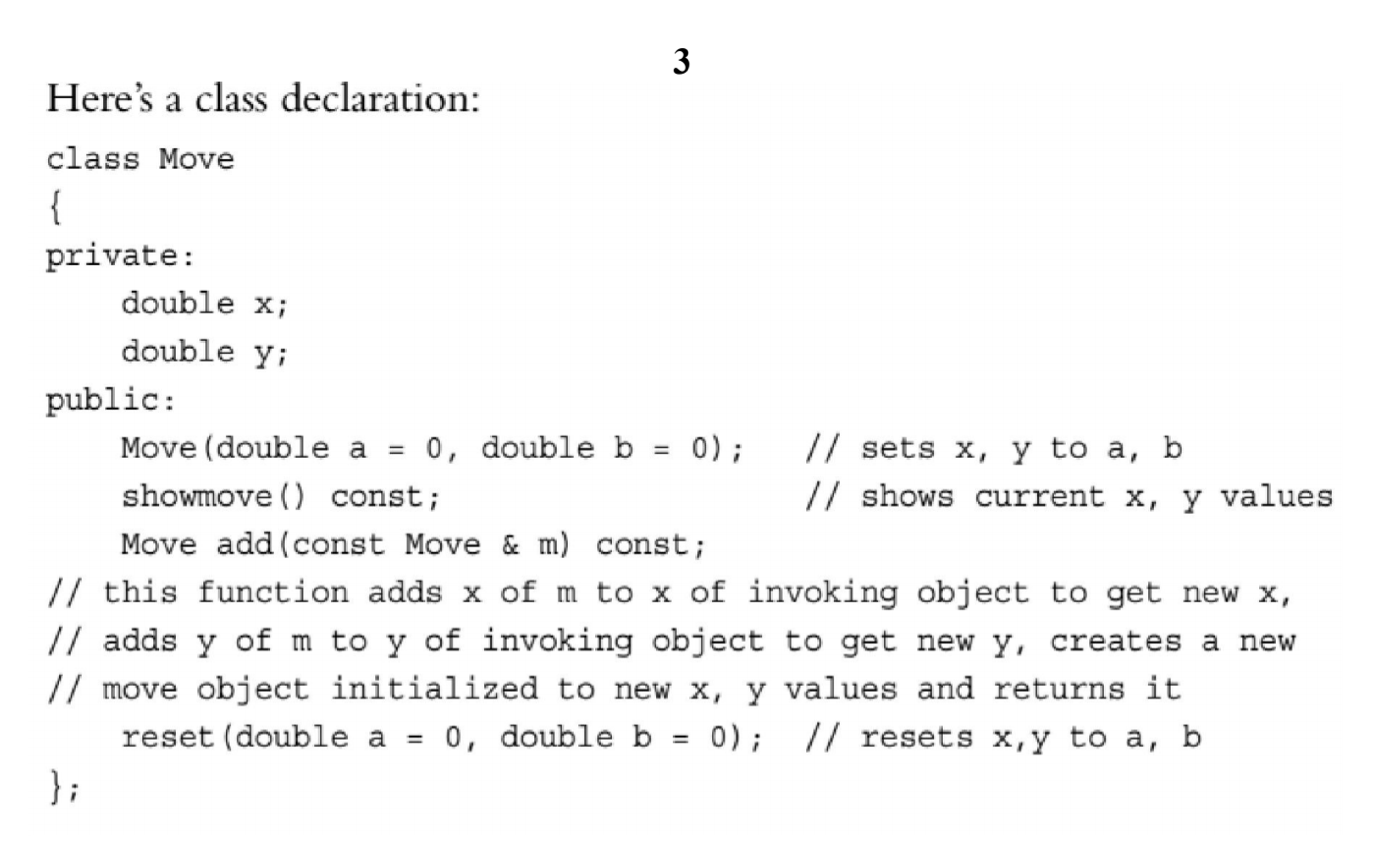
cout << "Done." << endl;

return 0;

}

Output:





#include <iostream>

#include <string>

using namespace std;

class Move

{

private:

double x;

double y;

public:

Move(double a = 0, double b = 0);

void showmove() const;

Move add(const Move &m) const;

void reset(double a = 0, double b = 0);

};

Move::Move(double a, double b)

{

this->x = a, this->y = b;

}

void Move::showmove() const

{

using std::cout;

using std::endl;

cout << "x: " << this->x << " y: " << this->y << endl;

}

Move Move::add(const Move &m) const

{

double newX = this->x + m.x;

double newY = this->y + m.y;

return Move(newX, newY);

}

void Move::reset(double a, double b)

{

this->x = a;

this->y = b;

}

void showmove(std::string, Move);

int main()

{

Move move1;

Move move2(3, 6);

Move move3(12);

showmove("move1 +++ ", move1);

showmove("move2 +++ ", move2);

showmove("move3 +++ ", move3);

Move move4 = move2.add(move3);

showmove("move4 +++ ", move4);

move4.reset();

showmove("move4 +++ ", move4);

move4.reset(-1);

showmove("move4 +++ ", move4);

move4.reset(-1, 2);

showmove("move4 +++ ", move4);

return 0;

}

void showmove(string aString, Move move)

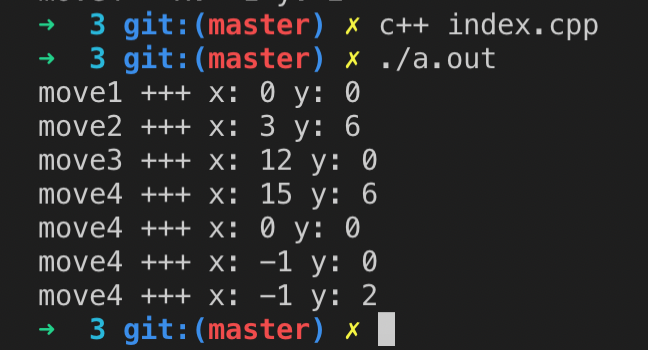
{

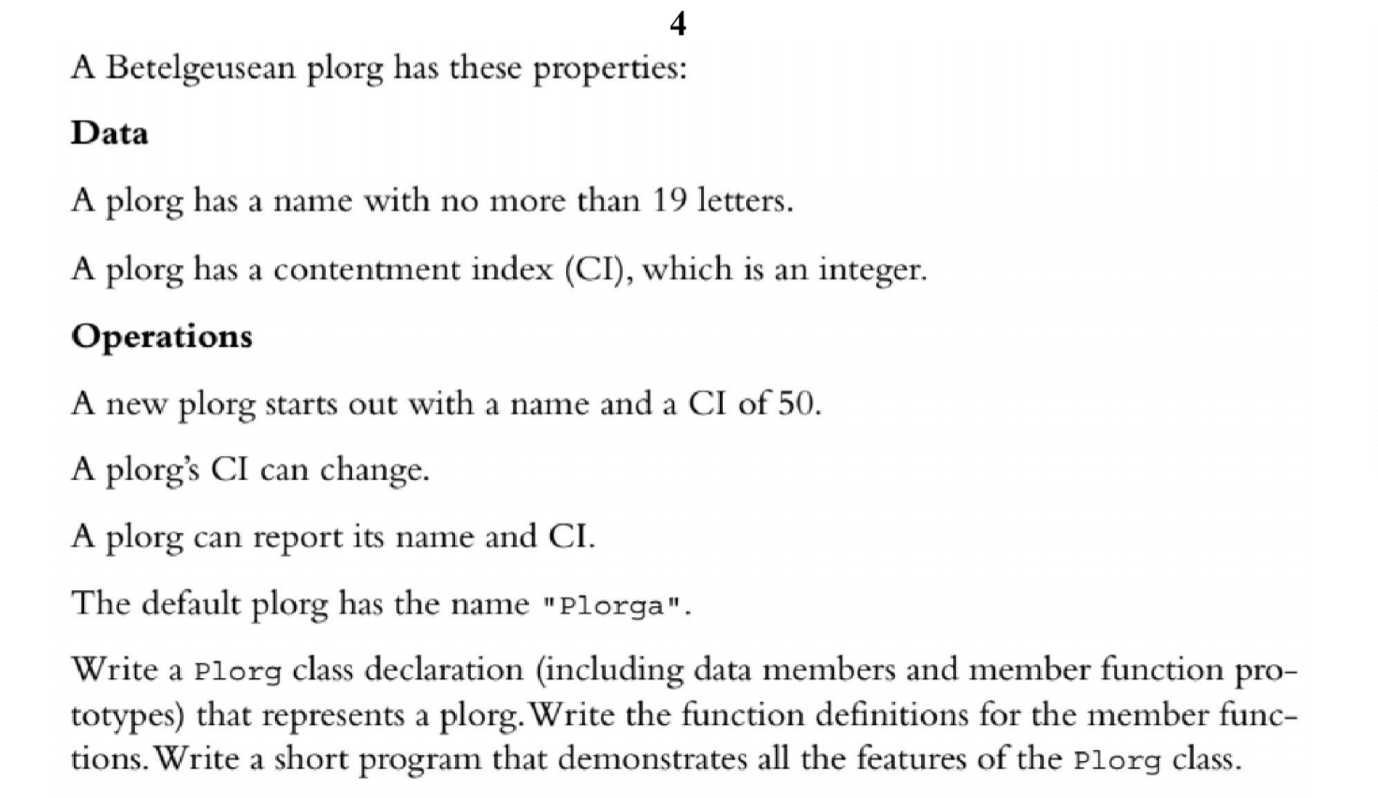
cout << aString;

move.showmove();

}

Output:





#include <iostream>

#include <cstring>

class Plorg {

static const int LEN = 19;

char name\_[LEN];

int cI\_;

public:

Plorg(const char\* name = "Plorga", int cI = 50);

void setCi(int newCi);

void showPlorg();

};

Plorg::Plorg(const char\* name, int cI) {

strncpy(this->name\_, name, this->LEN);

this->cI\_ = cI;

}

void Plorg::setCi(int newCi) {

this->cI\_ = newCi;

}

void Plorg::showPlorg() {

using std::cout;

using std::endl;

cout << "Name: " << this->name\_ << endl;

cout << "CI: " << this->cI\_ << endl;

}

int main() {

using std::cout;

Plorg plorg;

plorg.showPlorg();

plorg.setCi(1);

plorg.showPlorg();

plorg = Plorg("MyPlorg");

plorg.showPlorg();

plorg = Plorg("Another Plorg", 20);

plorg.showPlorg();

return 0;

}

Output:

