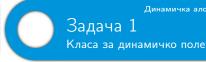


Аудиториски вежби 4

Динамичка алокација на меморија, композиција и наследување

Напреден развој на софтвер

- 1 Динамичка алокација на меморија
- 2 Композиција
- 3 Наследување



Да се напише класа за работа со еднодимензионални полиња од целобројни елементи. За полето се чуваат информации за вкупниот капцитет на полето, тековниот број на елементи. Резервацијата на меморијата да се врши динамички. Да се преоптоварат следните оператори

- [] за пристап до елемент и промена на вредноста на елемент од полето
- += за додавање нови броеви во полето и притоа ако е исполнет капацитетот на полето да се зголеми за 100%.

Да се напише главна програма каде ќе се инстанцира објект од класата и во него ќе се внесат N броеви од тастатура и потоа да се испечатат елементите на полето, неговиот капацитет и вкупниот број на елементи.

```
#include <iostream>
#include <cstdlib>
using namespace std;
class Array {
private:
   int *elem; // pokazuvac kon poleto
   int count; //kolku elementi ima
    int size; //kolku elementi moze da ima
public:
    Array(int size = 10) {
        this->size = size;
        count = 0:
        elem = new int[size]:
    Array(const Array &ob) {
        size = ob.size:
        count = ob.count;
        elem = new int[size];
        for (int i = 0: i < count: i++)
            elem[i] = ob.elem[i];
```

Задача 1 Решение 2/3

```
Array & operator = (Array & ob) {
    // Se sprechuva samododeluvanje
    if (this != &ob) {
        size = ob.size:
        count = ob.count;
        elem = new int[size];
        for (int i = 0: i < count: i++)
            elem[i] = ob.elem[i]:
    return *this:
Array & operator += (int el) {
    //ako ima kapacite seuste dodadi go clenot
    if (count < size) {
        elem[count++] = el;
    } else { //ako kapacitetot e popolnet
        int *tmp = new int[size * 2]://alociraj dvojno poveke memorija
        for (int i = 0; i < count; i++)</pre>
            tmp[i] = elem[i]:
        tmp[count++] = el;
        size *= 2;
        delete[] elem;
        elem = tmp;
    return *this;
~Array() {
    delete[] elem;
```

Задача 1

Решение 3/3

```
int &operator[](int i) {
        if (i >= 0 && i < count)
            return elem[i];
        else {
            cout << "Nadvor od opseg" << endl;
            exit(-1);
        }
    int getCapacity() {
        return size:
    int getCount() {
        return count:
};
int main() {
   Array pole;
   int n, tmp;
    cout << "Kolku elementi ke vnesete?" << endl:
   cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> tmp;
        pole += tmp:
    for (int i = 0; i < pole.getCount(); i++)</pre>
        cout << pole[i] << "\t";
    cout << endl << pole.getCapacity();
    return 0;
```

- 1 Динамичка алокација на меморија
- 2 Композиција
- 3 Наследување

Да се напише класа за работа со веб сервери (WebServer). За секој веб сервер се чува:

- неговото име (тах 30 знаци)
- листа од веб страници (динамичка низа објекти од класата WebPage).

За секоја веб страница се чува:

- url (max 100 знаци)
- содржина (динмички алоцирана низа од знаци).

За класата WebServer да се преотоварат операторите:

- += за додавање нова веб страница во серверот
- -= за бришење на веб страница од веб серверот.

Задача 2 Решение 1/5

```
#include <iostream>
#include <cstring>
using namespace std;
class WebPage {
private:
   char url[100]:
   char *sodrzina;
public:
    WebPage(char *url = "", char *sodrzina = "") {
        strcpy(this->url, url);
        this->sodrzina = new char[strlen(sodrzina) + 1];
        strcpv(this->sodrzina, sodrzina);
    WebPage(const WebPage& wp) {
        strcpy(this->url, wp.url);
        this->sodrzina = new char[strlen(wp.sodrzina) + 1]:
        strcpy(this->sodrzina, wp.sodrzina);
    ~WebPage() {
        delete[] sodrzina;
```

Задача 2

```
bool operator == (WebPage& wp) {
    return strcmp(url, wp.url) == 0;
}
WebPage operator = (WebPage& wp) {
    if (this != &wp) {
        strcpy(this -> vurl, wp.url);
        delete[] sodrzina;
        this -> sodrzina = new char[strlen(wp.sodrzina) + 1];
        strcpy(this -> sodrzina, wp.sodrzina);
}
return *this;
}
char *getUrl() {
    return url;
}
};
```

Задача 2 Решение 3/5

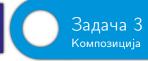
```
class WebServer {
private:
    char ime [30];
    int count:
    WebPage *wp;
public:
    WebServer(char *ime = "", int count = 0, WebPage *wp = 0) {
        strcpy(this->ime, ime);
        this -> count = count;
        this->wp = new WebPage[count];
        for (int i = 0; i < count; i++)
            this->wp[i] = wp[i];
    WebServer(const WebServer &ws) {
        strcpv(this->ime, ws.ime);
        this -> count = ws.count;
        this->wp = new WebPage[count];
        for (int i = 0: i < count: i++)
            this->wp[i] = ws.wp[i];
    WebServer& operator=(const WebServer &ws) {
        if (this != &ws) {
            strcpv(this->ime, ws.ime):
            this -> count = ws.count;
            delete [] this->wp;
            this->wp = new WebPage[count];
            for (int i = 0; i < count; i++)
                this->wp[i] = ws.wp[i];
        return *this;
```

Задача 2

```
~WebServer() {
    delete[] wp;
WebServer& operator += (WebPage webPage) {
    WebPage *tmp = new WebPage[count + 1];
    for (int i = 0; i < count; i++)
        tmp[i] = wp[i];
    tmp[count++] = webPage;
    delete[] wp;
    wp = tmp:
    return *this:
WebServer& operator -= (WebPage webPage) {
    int newCount = 0:
    WebPage *tmp = new WebPage[count];
    for (int i = 0; i < count; i++) {
        if (!(wp[i] == webPage)) {
            tmp[newCount++] = wp[i];
    delete[] wp:
    wp = tmp;
    count = newCount;
    return *this:
void listPages() {
    for (int i = 0: i < count: i++)</pre>
        cout << wp[i].getUrl() << endl;
```

}:

```
int main() {
    WebPage wp1("http://www.google.com", "The search engine");
    WebPage wp2("http://www.finki.ukim.mk", "FINKI");
    WebPage wp3("http://www.time.mk", "Site vesti");
    WebServer ws("Server");
    ws += wp1;
    ws += wp2;
    ws += up3;
    ws.listPages();
    ws -= wp3;
    ws.listPages();
    return 0;
}
```



Да се напише класа Datum, во која ќе се чуваат ден, месец и година (цели броеви).

Да се напише класа Vraboten, за кој се чува име (не повеќе од 100 знаци) и датум на раѓање (објект од Datum).

Да се напише класа Firma, во која се чува име на фирмата (не повеќе од 100 знаци) и низа од вработени (динамичи алоцирана низа од објекти од Vraboten). Да се преоптовари операторот += за додавање на вработен во фирмата. За оваа класа да се имплементираат метод кој ќе ги печати сите вработени во фирмата и метод кој ќе го пронајде и испечати најмладиот вработен.

```
#include <iostream>
#include <cstring>
using namespace std;
class Datum {
private:
   int den, mesec, godina;
public:
    Datum(int d = 1, int m = 1, int g = 1) {
        den = d:
        mesec = m:
        godina = g;
    bool operator < (Datum& datum) {
        if (godina != datum.godina)
            return godina < datum.godina;
        if (mesec != datum.mesec)
            return mesec < datum.mesec:
        return den < datum.den;
    friend ostream& operator << (ostream &out. Datum &datum) {
        out << datum.den << "." << datum.mesec << "." << datum.godina;
        return out;
};
```

```
class Vraboten {
private:
    char ime[100]:
    Datum datum_na_raganje;
public:
    Vraboten(char *i = "", int den = 0, int mesec = 0, int godina = 0) :
        datum_na_raganje(den, mesec, godina) {
        strcpy(ime, i);
    Datum& getDatum() {
        return datum_na_raganje;
    friend ostream& operator << (ostream &out, Vraboten &v) {
        out << v.ime << "\t" << v.datum na raganje:
        return out;
}:
```

```
class Firma {
private:
   char ime[100];
   Vraboten *vraboteni:
   int vkupno_vraboteni;
public:
    Firma(char *i = "") {
        vkupno vraboteni = 0:
        vraboteni = NULL;
        strcpv(ime, i):
    Firma(const Firma &f) {
        vkupno_vraboteni = f.vkupno_vraboteni;
        strcpy(ime, f.ime);
        vraboteni = new Vraboten[vkupno_vraboteni];
        for (int i = 0; i < vkupno_vraboteni; i++)</pre>
            vraboteni[i] = f.vraboteni[i];
    }
    Firma& operator=(const Firma &f) {
        if (this != &f) {
            vkupno_vraboteni = f.vkupno_vraboteni;
            strcpy(ime, f.ime);
            delete[] vraboteni:
            vraboteni = new Vraboten[vkupno_vraboteni];
            for (int i = 0; i < vkupno_vraboteni; i++)</pre>
                vraboteni[i] = f.vraboteni[i]:
        return *this:
```

```
~Firma() {
    delete[] vraboteni;
Firma& operator += (Vraboten v) {
    Vraboten *tmp = vraboteni;
    vraboteni = new Vraboten[vkupno_vraboteni + 1];
    for (int i = 0; i < vkupno_vraboteni; i++)</pre>
        vraboteni[i] = tmp[i]:
    vraboteni[vkupno_vraboteni++] = v;
    delete[] tmp;
    return *this;
void pecati() {
    cout << ime << endl:
    for (int i = 0; i < vkupno_vraboteni; i++)</pre>
        cout << vraboteni[i] << endl;
}
void pecatiNajmlad() {
    Vraboten najmlad = vraboteni[0];
    for (int i = 1; i < vkupno_vraboteni; i++)</pre>
        if (najmlad.getDatum() < vraboteni[i].getDatum())</pre>
            najmlad = vraboteni[i];
    cout << "Naimlad vraboten e : " << endl:
    cout << najmlad;
```

}:

```
int main() {
    Vraboten v1("Petko", 20, 2, 1990);
    Vraboten v2("Trajko", 3, 7, 1992);
    Vraboten v3("Mitko", 12, 9, 1989);
    Firma firma("Finki");
    firma += v1;
    firma += v2;
    firma += v3;
    firma pecati();
    cout << endl;
    firma.pecatiNajmlad();
    cout << endl;
    return 0;
}</pre>
```

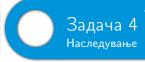
- 1 Динамичка алокација на меморија
- 2 Композиција
- 3 Наследување

Наследување

```
class ime_izvedena : pristap ime_osnovna {
    // definicija i implementacija na klasata
};
```

Пример наследување со public пристап

```
class Osnovna {
private:
    int a;
public:
    Osnovna(int a) { this->a = a; }
    Osnovna() {}
}
class Izvedena : public Osnovna {
private:
    int b;
public:
    Izvedena(int a, int b) : Osnovna(a) {
        this->b = b;
    }
}
```



Да се напише класа за точка во дводимензионален (2Д) координатен систем. Од оваа класа да се изведе класа за точка во тродимензионален (3Д) кооридинатен систем.



```
#include <iostream>
using namespace std;
class Point {
private:
   int x, y;
public:
    Point(int xx = 0, int vv = 0) {
        cout << "Constructor Point" << endl;</pre>
        x = xx;
        y = yy;
    "Point() { cout << "Destructor Point" << endl; }
   int getX() { return x: }
    int getY() { return y; }
    void setX(int xx) { x = xx; }
   void setY(int vv) { v = vv; }
}:
class Point3D : public Point {
private:
   int z:
public:
    Point3D(int x = 0, int y = 0, int z = 0): Point(x, y) {
        cout << "Constructor Point3D" << endl:
       this ->z = z;
```



Да се напише класа која ќе опишува банкарска сметка. За сметката се чуваат информации за бројот, името на сопственикот, како и салдото (моменталната состојба). За оваа класа да се имплементираат методи за додавање и повлекување средства од сметката, како метод кој пачати на екран преглед на сметката. Доколку сопственикот нема доволно средства при повлекувањето се печати соодветна порака. Од оваа класа да се изведе класа за банкарска сметка со дозволен максимален заем на кој се пресметува соодветна камата. За оваа класа да се преоптоварат методите за повлекување средства и печатање преглед на сметката.

```
#include <iostream>
#include <cstring>
using namespace std;

class Smetka {
    private:
        char ime[100];
        long broj;
        double saldo;

public:
        Smetka(char *i = "", long b = 0, double s = 0) {
            strcpy(ime, i);
            broj = b;
            saldo = s;
        }
}
```

Задача 5 Решение 2/3

```
void deposit(double suma) {
        saldo += suma;
    void podiganje(double suma) {
        if (suma > saldo) {
            cout << "Ne moze da podignete " << suma << " denari.";</pre>
        } else {
            saldo -= suma;
    double getSaldo() {
        return saldo;
    void pregled() {
        cout << "Ime: " << ime << endl;
        cout << "Broj na smetka: " << broj << endl;
        cout << "Saldo: " << saldo << endl;
};
```



```
class SmetkaPlus: public Smetka {
private:
   double maxZaem;
   double kamata:
   double dolzi;
public:
    SmetkaPlus(char *i = "", long b = 0, double s = 0, double m = 0, double k =
            0. double d = 0):
        Smetka(i, b, s) {
        maxZaem = m:
        kamata = k:
        dolzi = d;
    void podiganje(double suma) {
        if (suma <= getSaldo()) {</pre>
            Smetka::podiganje(suma);
        } else if (suma < getSaldo() + maxZaem - dolzi) {</pre>
            double advance = suma * (1.0 + kamata);
            deposit(advance);
            Smetka::podiganje(suma):
        } else {
            cout << "Ne moze da podignete sredstva!" << endl;</pre>
    void pregled() {
        Smetka::pregled();
        cout << "Dolzi: " << dolzi << endl:
};
```

```
int main() {
    Smetka Petko("Petko Petkovski", 381299, 4000);
    SmetkaPlus Trpe("Trpe Trpevski", 382288, 3000);
    Petko.pregled();
    Trpe.pregled();
    cout << "Deposit od 1000 den na smetkata na Trpe:\n";
    Trpe.deposit(1000);
    cout << "Novo saldo: " << Trpe.getSaldo() << endl;
    cout << "Podiganje na 4200 den od smetkata na Petko:\n";
    Petko.podiganje(4200);
    cout << "Saldoto na Petko e: " << Petko.getSaldo() << endl;
    cout << "Saldoto na Petko e: " << Petko.getSaldo() << endl;
    cout << "Odiganje na 4200 den od smetkata na Trpe:\n";
    Trpe.podiganje(4200);
    Trpe.podiganje(4200);
    Trpe.pregled();
    return 0;
}</pre>
```

Предавања, аудиториски вежби, соопштенија courses.finki.ukim.mk

Изворен код на сите примери и задачи bitbucket.org/tdelev/finki-nrs

Прашања и одговори qa.finki.ukim.mk