

Автор: Білий Вадим., КІТ-119а

Дата: 1.05.2020

## Лабораторна робота 5. АГРЕГАЦІЯ ТА КОМПОЗИЦІЯ

Тема. Класи. Агрегація. Композиція. Ключові слова typedef та auto.

Мета – отримати поняття агрегація та композиція; отримати знання про призначення ключових слів typedef та auto.

Загальне завдання

Дослідити заздалегідь визначені типи даних з бібліотеки `<cstdlib>` / `<stddef.h>`.

Модернізувати розроблені у попередній роботі класи таким чином: - замінити типи даних, що використовуються при індексуванні на типи з указаної бібліотеки;

- створити власний синонім типу, визначивши його необхідність;

- створити / оновити функцію сортування масиву, де крім поля, по якому виконується сортування, передається і вказівник на функцію, яка визначає напрям сортування;

- у базовий клас додати два поля, що мають кастомний тип даних (тип даних користувача) та які будуть відображати відношення «агрегація» та «композиція», при цьому оновити методи читання та запису об'єкта;

- ввести використання ключового слова auto як специфікатор зберігання типу змінної. Визначити плюси та мінуси цього використання.

Опис класів

Базовий клас: `Ccooperator`

Клас, що має в собі масив базового класу та методи для роботи з ним: `CList`

Клас що демонструє агрегацію: `Schpi`

Клас що демонструє композицію: `Cweight`

## Опис змінних

const char\* name – ім'я.

int amount - кількість елементів

Ccooperator\* fEl - 1 масив

Ccooperator\* fEl1 - 2 масив

int id - Id персони

int age - вік

int salary -заробітна плата

Cweight weight– вага

Cchpi kaf– інформація про ректора

## Опис методів

void setId(const int id); - встановлює id.

void setAge(const int age);- встановлює вік.

void setSalary(const int salary); - встановлює заробітну плату.

int getId()const; - повертає id.

int getAge()const; - повертає вік.

int getSalary()const; - повертає заробітну плату.

string getKaf() – повертає інформацію про ректора

Cchpi\* getAdresKaf(); - повертає адрес на агрегатний об'єкт

int getWeight(); - повертає вагу

Ccooperator(); - конструктор.

Ccooperator(int a, int b, int c, const char\* d); - конструктор с параметрами.

Ccooperator(const Ccooperator& a) – конструктор копіювання.

~Ccooperator() { } – деструктор.

int averageSalary() – середня заробітна плата.

void creatMass(int a); - створює масив.

void Add(cooperator); - додає елемент.

void Delete(int ); - видаляє елемент.

cooperator getCooperator(int a); - повертає елемент.

void showAll(); - показує всі елементи.

cooperator findCooperator(const int a); - знаходить елемент.

int getAmount(); - повертає кількість елементів.

void End(); - видаляє всі масиви.

void ListK::Creatmass(int n); - створює масив

CcoopK& ListK::getCcoopK(int n); - повертає елемент масиву

## Текст програми

### Cooperator.h

```
#pragma once
#include <sstream>
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "Cchpi.h"
#include "Cweight.h"
#include <locale>
using namespace std;
class Ccooperator
{
private:
    int id, age, salary;
    string name;
    Cchpi* kaf;
    Cweight weight;

public:
    void setName(string name);
    void setId(const int id);
    void setAge(const int age);
    void setSalary(const int salary);
    void setKaf(Cchpi* kaff);
    void setWeight(const int w);
    string getName();
    int getId()const;
    int getAge()const;
    int getSalary()const;
    string getKaf();
    Cchpi* getAdresKaf();
    int getWeight();
    Ccooperator();
    Ccooperator(int a, int b, int c, int d, Cchpi* i, const string f);
    Ccooperator(const Ccooperator& a);
    ~Ccooperator() {};
};
```

### Cooperator.cpp

```
#include "cooperator.h"
void Ccooperator::setName(string name) {
    this->name = name;
}
```

```

string Ccooperator::getName() {
    return this->name;
}
void Ccooperator::setId(const int id) {
    this->id = id;
}
void Ccooperator::setAge(const int age) {
    this->age = age;
}
void Ccooperator::setKaf(Cchpi* kaff){
    this->kaf=kaff;
}
void Ccooperator::setWeight(const int w){
    weight.setWeigt(w);
}
string Ccooperator::getKaf() {
    return kaf->getName();
}
Cchpi* Ccooperator::getAdresKaf()
{
    return kaf;
}
int Ccooperator::getWeight() {
    return weight.getWeigt();
}
void Ccooperator::setSalary(const int salary) {
    this->salary = salary;
}
int Ccooperator::getId()const {
    return this->id;
}
int Ccooperator::getAge()const {
    return this->age;
}
int Ccooperator::getSalary()const {
    return this->salary;
}
Ccooperator::Ccooperator() :id(0), age(0), salary(0), weight(10),kaf(NULL),name("Ivan") {
    cout << "\nБыл вызван конструктор по умолчанию в объекте с id: " << id<<"\n";
}
Ccooperator::Ccooperator(const Ccooperator& a) :id(a.id), age(a.age), salary(a.salary),
weight(a.weight), kaf(a.kaf), name(a.name) {
    cout << "\nБыл вызван конструктор по умолчанию в объекте с id: " << id<<"\n";
};
Ccooperator::Ccooperator(int a , int b , int c ,int d,Cchpi* i ,const string f) : id(a),
age(b), salary(c), weight(d),kaf(i),name(f) {
    cout << "\nБыл вызван конструктор по умолчанию в объекте с id: " << id<<"\n";
};

```

## List.h

```

#pragma once
#include "cooperator.h"
#include <regex>
typedef bool (Fun)(const int&,const int &);
class Clist {
private:
    int amount;
    Ccooperator* fEl;
    Ccooperator* fEl1;
public:
    static bool sortAsc(const int& a, const int& b);
    static bool sortDesc(const int& a, const int& b);
    void Sort(string sprz, Fun s);
    void twoWorld();

```

```

    int averageSalary();
    void writeToFile(string fileName);
    void readFromFile(string fileName);
    void creatMass(int a);
    //Ccooperator creatEl1();
    Ccooperator creatEl2();
    void Add(Ccooperator);
    void Delete(int b);
    void AddWhithString();
    string findCooperator(const int a);
    Ccooperator getCooperator(int a);
    void showAll();
    int getAmount();
    void End();
};

```

## List.cpp

```

#include "list.h"
#include <sstream>
#include <iostream>
#include "list.h"
#include <sstream>
#include <iostream>
#include <fstream>

bool CList::sortAsc(const int &a, const int &b) {
    return a > b;
}

bool CList::sortDesc(const int &a, const int &b) {
    return a < b;
}

void CList::Sort(string sprz, Fun s) {
    int prz=0;
    Ccooperator temp;
    if (sprz == "id") {
        do {
            prz = 0;
            for (size_t i = 1; i < amount; i++) {
                if (s(fEl[i - 1].getId(), fEl[i].getId())) {
                    temp = fEl[i - 1];
                    fEl[i - 1] = fEl[i];
                    fEl[i] = temp;
                    prz = 1;
                }
            }
        } while(prz!=0);
    }
    if(sprz == "salary"){
        do {
            prz = 0;
            for (size_t i = 1; i < amount; i++) {
                if (s(fEl[i - 1].getSalary(), fEl[i].getSalary())) {
                    temp = fEl[i - 1];
                    fEl[i - 1] = fEl[i];
                    fEl[i] = temp;
                    prz = 1;
                }
            }
        } while (prz != 0);
    }
    if (sprz=="age") {
        do {
            prz = 0;

```

```

        for (size_t i = 1; i < amount; i++) {
            if (s(fEl[i - 1].getAge(), fEl[i].getAge())) {
                temp = fEl[i - 1];
                fEl[i - 1] = fEl[i];
                fEl[i] = temp;
                prz = 1;
            }
        } while (prz != 0);
    }
}

void CList::creatMass(int a)
{
    amount = a;
    fEl = new Ccooperator[amount];
    for (std::size_t i = 0; i < amount; i++) {
        fEl[i] = creatEl2();
    }
}

string CList::findCooperator(const int a) {
    std::stringstream ss;
    string ab;
    ss << " ";
    int b = -1, count = 0;
    for (std::size_t i = 0; i < amount; i++) {
        if (a == fEl[i].getId()) {
            count++;
            b = i;
        }
    }
    if (count >= 1) {
        cout << "Есть " << count << " похожих элементов, будет возвращен последний элемент";
        ss << "\nId: " << fEl[b].getId() << "\nAge:" << fEl[b].getAge() <<
        "\nSalary: " << fEl[b].getSalary() << "\nName " << fEl[b].getName();
        ab = ss.str();
        return ab;
    }
    if (count == 0) {
        cout << "Похожих элементов нет, возвращен пустой символ";
        return ab;
    }
}

int CList::averageSalary() {
    auto averageSalary = 0;
    for (std::size_t i = 0; i < amount; i++)
    {
        averageSalary = averageSalary + fEl[i].getSalary();
    }
    return averageSalary = averageSalary / amount;
};

Ccooperator CList::creatEl2() {
    Ccooperator El;
    El.setId(0);
    El.setSalary(0);
    El.setAge(0);
    return El;
}

void CList::readFromFile(string fileName) {
    End();
    creatMass(0);
    int integer;
    string line;
    string str;

```

```

string tstr="";
Ccooperator temp;
ifstream file;
file.open(fileName);
if (!file.is_open())
{
    cout << " Файл не открыт, давай по новой Миша\n";
    return;
}
std::regex regular_main(
"([0-9]*[ \\f\\n\\r\\t\\v]{1})"
"([0-9]*[ \\f\\n\\r\\t\\v]{1})"
"([0-9]*[ \\f\\n\\r\\t\\v]{1})"
"([0-9]*)"
"([ \\f\\n\\r\\t\\v]{1}[A-Z A-Я]{1}[a-z а-я 0-9]*)"
"([ \\f\\n\\r\\t\\v]{1}[A-Z A-Я]{1}[a-z а-я 0-9]*[\\.\\,\\;\\:~]{0,1})([
\\f\\n\\r\\t\\v]{0,1}[a-z а-я 0-9]*[\\.\\,\\;\\:~]{0,1}[ \\f\\n\\r\\t\\v]{0,1})*"
);
std::cmatch result;
std::stringstream ss;
Cchpi* a;
while (getline(file, line)) {

    if (regex_match(line.c_str(), result, regular_main)) {
        ss << result[1];
        ss >> integer;
        temp.setId(integer);
        ss.clear();
        ss << result[2];
        ss >> integer;
        temp.setAge(integer);
        ss.clear();
        ss << result[3];
        ss >> integer;
        temp.setSalary(integer);
        ss << result[4];
        ss >> integer;
        temp.setWeight(integer);
        ss.clear();
        ss << result[5];
        ss >> str;
        a = new Cchpi;
        a->setName(str);
        temp.setKaf(a);
        for (std::size_t i = 6; i < result.size(); i++)
        {
            ss.clear();
            ss << result[i];
            ss >> str;
            if ((str+" ")!=tstr)
                tstr = tstr + str + " ";
        }

        temp.setName(tstr);
        tstr = "";
        Add(temp);
    }
    result.end();
}
file.close();
}

void Clist::twoWorld() {
    std::regex regular("([A-Z A-Я a-z а-я 0-9]+[\\.\\,\\;\\:~]{0,1}[
\\f\\n\\r\\t\\v]{1}[A-Z A-Я a-z а-я 0-9]+[\\.\\,\\;\\:~]{0,1}[ \\f\\n\\r\\t\\v]{0,1}){1}");
    std::cmatch result;
    for(std::size_t i=0;i<amount;i++)

```

```

        if (regex_match(fEl[i].getName(), regular)) {
            cout << "\n" << "ID:" << fEl[i].getId() << "\n Age: " <<
fEl[i].getAge() << "\n Salary: " << fEl[i].getSalary() << "\n Name: " <<
fEl[i].getName();
        }
    }
}

void CList::writeToFile(string fileName) {
    ofstream file;
    string str1;
    std::stringstream ss;
    file.open(fileName);
    if (!file.is_open())
    {
        cout << " Файл не открыт, давай по новой Миша\n";
        return;
    }
    for (std::size_t i = 0; i < amount; i++) {
        file << fEl[i].getId() << " " << fEl[i].getAge() << " " <<
fEl[i].getSalary() << " " << fEl[i].getName() << " " << fEl[i].getWeight() << " " <<
fEl[i].getKaf() << "\n";
    }

    file.close();
}

void CList::AddWhithString() {
    Ccooperator temp;
    std::stringstream ss1;
    fEl1 = new Ccooperator[amount + 1];
    for (std::size_t i = 0; i < amount; i++) {
        fEl1[i] = fEl[i];
    }

    std::cout << "\nВведите данные с клавиатуры в таком порядке: id, age, salary,
name,\n";
    string tid = " ", tage = " ", tsalary = " ", tweight=" ", tkaf = " ", tname = " ";
    int tid1;
    string tname1 = " ";
    cin >> tid >> tage >> tsalary >> tweight >> tkaf >> tname ;
    ss1 << tid;
    ss1 >> tid1;
    temp.setId(tid1);
    ss1.clear();
    ss1 << tage;
    ss1 >> tid1;
    temp.setSalary(tid1);
    ss1.clear();
    ss1 << tsalary;
    ss1 >> tid1;
    temp.setAge(tid1);
    ss1.clear();
    ss1 << tname;
    ss1 >> tname1;
    temp.setName(tname1);
    ss1.clear();
    ss1 << tweight;
    ss1 >> tid1;
    temp.setWeight(tid1);
    ss1.clear();
    ss1 << tkaf;
    ss1 >> tname;
    Cchpi* a=new Cchpi;
    a->setName(tname);
    temp.setKaf(a);
    fEl1[amount] = temp;
    delete[] fEl;
}

```



```

        amount++;
        fEl = new Ccooperator[amount];
        for (std::size_t i = 0; i < amount; i++) {
            fEl[i] = fEl1[i];
        }
        delete[] fEl1;
    }
    void CList::Add(Ccooperator El1) {
        fEl1 = new Ccooperator[amount + 1];
        for (std::size_t i = 0; i < amount; i++) {
            fEl1[i] = fEl[i];
        }
        fEl1[amount] = El1;
        delete[] fEl;
        amount++;
        fEl = new Ccooperator[amount];
        for (int i = 0; i < amount; i++) {
            fEl[i] = fEl1[i];
        }
        delete[] fEl1;
    }
    int CList::getAmount() {
        return amount;
    }
    void CList::Delete(int a) {
        Ccooperator* fEl1 = new Ccooperator[amount - 1];
        for (std::size_t i = 0; i < a - 1; i++) {
            fEl1[i] = fEl[i];
        }
        for (std::size_t i = a - 1, j = a; j < amount; i++, j++) {
            fEl1[i] = fEl[j];
        }
        delete[] fEl;
        amount--;
        fEl = new Ccooperator[amount];
        for (std::size_t i = 0; i < amount; i++) {
            fEl[i] = fEl1[i];
        }
        delete[] fEl1;
    }
    Ccooperator CList::getCooperator(const int a) {
        return fEl[a];
    }
    void CList::showAll() {
        for (std::size_t i = 0; i < amount; i++) {
            cout << "ID:" << getCooperator(i).getId() << "\n Age: " <<
            getCooperator(i).getAge() << "\n Salary: " << getCooperator(i).getSalary() << "\n
            Weight:" << getCooperator(i).getWeight() << "\n Name Rector:" << getCooperator(i).getKaf() <<
            "\n Name: " << getCooperator(i).getName();
        }
    }
    void CList::End() {
        for (int i = 0; i < amount; i++) {
            if(fEl1[i].getAdresKaf())
                delete fEl[i].getAdresKaf();
        }
        delete[] fEl;
    }
}

```

## Cchpi.h

```

#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
class Cchpi

```

```

{
private:
    std::string nameRect;
    int ageRect;
public:
    std::string getName()const;
    int getAge() const;
    void setName(std::string name);
    void setAge(int age);
    Cchpi();
    Cchpi(std::string name, int age);
    Cchpi(const Cchpi& temp);
    ~Cchpi();
};

```

## Cchpi.cpp

```

#include "Cchpi.h"
#include <iostream>
std::string Cchpi::getName()const {
    return nameRect;
}
int Cchpi::getAge() const
{
    return ageRect;
}
void Cchpi::setName(std::string name) {
    this->nameRect = name;
}
void Cchpi::setAge(int age)
{
    this->ageRect = age;
}
Cchpi::Cchpi() : nameRect("E. Sokon"), ageRect(68) {};
Cchpi::Cchpi(std::string name, int age) : nameRect(name), ageRect(age) {};
Cchpi::Cchpi(const Cchpi& temp) : nameRect(temp.nameRect), ageRect(temp.ageRect) {};
Cchpi::~Cchpi() {};

```

## Cweight.h

```

#pragma once
#include <iostream>
class Cweight
{
private:
    int weight;
public:
    void setWeigt(const int m);
    int getWeigt() const;
    Cweight();
    Cweight(const int m);
    Cweight(const Cweight& m);
    ~Cweight();
};

```

## Cweight.cpp

```

#include "Cweight.h"

```

```

void Cweight::setWeigt(const int m) {
    weight = m;
}
int Cweight::getWeigt() const {
    return weight;
}
Cweight::Cweight() : weight(0){}
Cweight::Cweight(const int m): weight(m){}
Cweight::Cweight(const Cweight& m): weight(m.weight){}
Cweight::~~Cweight(){}

```

## Test.cpp

```

#include "cooperator.h"
#include "list.h"
#define _CRT_SECURE_NO_WARNINGS
#define N 5
#include <iostream>
#include <clocale>
using namespace std;
int main() {
    setlocale(LC_ALL, "rus");
    Ccooperator a;
    a.setAge(0);
    a.setAge(0);
    a.setSalary(0);
    CList a1[N];
    int test[N];
    int rezult1[N];
    int rezult2[N];
    int rezult3[N];
    test[0] = 1;
    test[1] = 5;
    test[2] = 10;
    test[3] = 25;
    test[4] = 50;
    rezult1[0] = 1;
    rezult1[1] = 5;
    rezult1[2] = 10;
    rezult1[3] = 25;
    rezult1[4] = 50;
    rezult2[0] = 2;
    rezult2[1] = 6;
    rezult2[2] = 11;
    rezult2[3] = 26;
    rezult2[4] = 51;
    rezult3[0] = 1;
    rezult3[1] = 5;
    rezult3[2] = 10;
    rezult3[3] = 25;
    rezult3[4] = 50;
    for (int i = 0; i < N; i++) {
        a1[i].creatMass(test[i]);
        if (a1[i].getAmount() == rezult1[i]) {
            printf("Тест 1.%i пройден\n", i);
        }
        else {
            printf("Тест 1.%i не пройден\n", i);
        }
    }
    for (int i = 0; i < N; i++) {
        a1[i].Add(a);
        if (a1[i].getAmount() == rezult2[i]) {

```

```

        printf("Тест 2.%i пройден\n", i);
    }
    else {
        printf("Тест 2.%i не пройден\n", i);
    }
}
for (int i = 0; i < N; i++) {
    a1[i].Delete(test[i]);
    if (a1[i].getAmount() == rezult3[i]) {
        printf("Тест 3.%i пройден\n", i);
    }
    else {
        printf("Тест 3.%i не пройден\n", i);
    }
}
}
Ccooperator Obtest1, Obtest2;
Obtest1.setSalary(-200);
Obtest2.setSalary(300);
CList TestList;
TestList.creatMass(0);
TestList.Add(Obtest1);
TestList.Add(Obtest2);
if (TestList.averageSalary() == 50) {
    printf("Тест 4 пройден\n");
}
else {
    printf("Тест 4 не пройден\n");
}
}
}

```

### Висновок

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з агрегацією та композицією.

Було розроблено програму, що демонструє агрегацію та композицію. Створено агрегатний та композитний клас.

Агрегація – це коли клас є полем іншого класу, та не обов’язково знищується одночасно з головним класом.

Композиція – це коли клас є полем іншого класу, та обов’язково знищується одночасно з головним класом.

Програма протестована, витоків пам’яті немає, виконується без помилок.