Документация на курсов проект №12:
Програма за обработка на цветя

Георги Миленов Соколов КСТ 3б група ФН:21621397

Условие:

I. Да се състави клас за цветя CFlowers с член променливи: Име на цветето, цъфтящо (да/не), семейство, години (едногодишно, двугодишно, многогодишно).
Съставете функции за:

1. за установяване на член променливите
2. функция за извеждане в изходен поток (отделно файл, отделно конзола)

Съставете конструктори:

1. подразбиращ се конструктор
2. копиращ конструктор
3. експлицитен конструктор

Съставете функции за:

1. Защита от невалидни данни
2. Определя дали в името на цветето се съдържа словосъчетание от името на семейството (пр. папрат папратови)
3. Търсене по семейство
4. Оператор за присвояване =
5. Логически оператор, който проверява дали цветето е цъфтящо
6. Оператор за съвпадение (по семейство).

II. Да се състави клас CParnik, който съдържа CFlowers с член променливи име на парника, брой цветя, брой приходи за една година.
Съставете следните функции:

1. Създаване на обект чрез друг обект
2. Създаване на обект чрез експлицитно зададени параметри
3. Функция, определяща броя цветя дали надвишава 1000.

III. Съставете главна програма, илюстрираща използването на създадените класове. Да се създаде контейнер вектор от 10 обекта от класа CParnik и да се направят следните справки, като:

1. извеждане в отделен списък (list) на цветята едногодишни, срещащи се най-малко в 3 парника
2. Функция, определяща от всяко семейство колко е броя на цветята.
3. да се изведат в отделни файлове всички парници, които отглеждат само цъфтящи с брой приходи над 100000 лева на година
4. да се изведат цветята, които са над 20 за години, чийто тип е подаден като параметър
5. да се изведат всички данни за парниците, с брой приходи, получени като параметър от текстов файл
6. при подаден като параметър общ брой цветя, да се изведат парниците, които са с приходи от първата тройка в отделен контейнер.

IV. Документиране на курсовия проект: .doc файл със заглавна страница, условие на задачата, кратко описание на класовете и функциите и листинг на програмата с коментари.

Описание класовете и функциите им:

***енумератор FlowerType** – определя 3-те възможни вида цветя (едногодишни, двугодишни, многогодишни).*

клас CFlowers:

Член променливи:

1) string flowerName - *име на цветето;*
2) bool isFlowering - *цъфтящо (да/не);*
3) string flowerFamily - *семейство;*
4) FlowerType flowerType - *години (едногодишно, двугодишно, многогодишно).*

Конструктори:

1) CFlowers() - *подразбиращ се конструктор;*
2) CFlowers(const CFlowers& fl) - *копиращ конструктор;*
3) CFlowers(const string& name, bool isF, const string& fam, const FlowerType& type) - *експлицитен конструктор;*
4) CFlowers(istream& flowerInfo) - *експлицитен конструктор, приемащ информация от входен поток.*

Функции:

1) void setFlowerName(const string& name) - *за установяване на* flowerName;
2) void setIsFlowering(bool isF) - *за установяване на* isFlowering;
3) void setFlowerFamily(const string& fam) - *за установяване на* flowerFamily;
4) void setFlowerType(const FlowerType& type) - *за установяване на* flowerType;
5) void consoleOutput() - *за извеждане в конзола;*
6) void fileOutput(const string& filePath) - *за извеждане във файл;*
7) bool isNameValid(const string& flowerName) - *за защита от невалидни данни;*
8) bool isFloweringValid(const string& isFlowering) - *за защита от невалидни данни;*
9) bool isFamilyValid(const string& flowerFamily) - *за защита от невалидни данни;*
10) bool isTypeValid(const string& flowerType) - *за защита от невалидни данни;*
11) bool isFamilyInName() - *определя дали в името на цветето се съдържа словосъчетание от името на семейството (пр. папрат папратови);*
12) bool sameFamily(const string& fam) - *за търсене по семейство;*
13) void operator=(const CFlowers& fl) - *оператор за присвояване =;*
14) bool operator>(const CFlowers& dummy) - *логически оператор, който проверява дали цветето е цъфтящо;*
15) bool operator==(const CFlowers& fl) - *оператор за съвпадение (по семейство);*
16) string getFlowerName() - *връща* flowerName*;*
17) string getFlowerFamily() - *връща* flowerFamily;
18) FlowerType getFlowerType() - *връща* flowerType;
19) friend ostream& operator<<(ostream& toStream, const CFlowers& fl) - *приятелски глобален оператор за извеждане на цвете в изходен поток.*

клас Cparnik:

Член променливи:

1) vector<CFlowers> flowersInside - *вектор, който съдържа CFlowers;*
2) string parnikName - *име на парника;*
3) int flowerCount - *брой цветя;*
4) int yearlyIncome - *брой приходи за една година.*

Конструктори:

5) CParnik() - *подразбиращ се конструктор;*
6) CParnik(const CParnik& p) - *за създаване на обект чрез друг обект;*
7) CParnik(const string& pn, int fc, int yi) - *за създаване на обект чрез експлицитно зададени параметри;*
8) CParnik(const string& filePath) - *за създаване на обект чрез експлицитно зададен като параметър път към файл.*

Функции:

1) bool areFlowersMoreThanA1000() - *определяща броя цветя дали надвишава 1000;*
2) void setParnikName(const string& n) - *за задаване на* parnikName;
3) bool allFlowering() - *за проверка дали всички цветя в парника цъфтят;*
4) int getYearlyIncome() - *връща* yearlyIncome*;*
5) int getFlowerCount() - *връща* flowerCount;
6) vector<CFlowers>& getFlowersInside() - *връща референция към* flowersInside;
7) bool operator<(const CParnik& p) - *сравнява парници по* yearlyIncome;
8) bool operator==(const CParnik p) - *сравнява парници по* yearlyIncome;
9) friend ostream& operator<<(ostream& toStream, const CParnik& p) - *приятелски глобален оператор за извеждане на всички данни за парник в изходен поток.*

клас MainClass:

Член променлива:

- vector<CParnik> ps - *вектор от парници;*

Конструктор:

- MainClass() - *създава 10 парника в ps чрез конструкторите на* Cparnik;

Функции:

1) list<CFlowers> getFlowersInAtLeast3P() - *извежда в списък цветята едногодишни, срещащи се най-малко в 3 парника;*
2) void outputFamiliesWithCount() - *определя от всяко семейство колко е броя на цветята;*
3) void outputFloweringParniciOver100000() - *извежда в отделни файлове всички парници, които отглеждат само цъфтящи с приходи над 100000 лева;*
4) void outputFlowersOver20ByType(const FlowerType& t) - *извежда цветята, които са над 20 за години, чийто тип е подаден като параметър;*
5) void outputByIncomeFromFile(const string& filePath) - *извежда всички данни за парниците, с брой приходи, получени като параметър от текстов файл;*
6) vector<CParnik> getRichestParnici(int& flowersInAll3) - *при подаден като параметър общ брой цветя, да се извеждат парниците, които са с приходи от първата тройка във вектор.*

функция main() - *тества функциите на* ***MainClass***.

Код на задачата:

```
#include <algorithm>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <list>

using namespace std;

enum FlowerType{
    annual, biennial, perennial
};
/*
I. Да се състави клас за цветя CFlowers с член променливи:
Име на цветето, цъфтящо (да/не), семейство, години (едногодишно, двугодишно,
многогодишно).
*/
class CFlowers{
    string flowerName;
    bool isFlowering;
    string flowerFamily;
    FlowerType flowerType;
    public:
    /*
    Съставете конструктори:
        1. подразбиращ се конструктор
        2. копиращ конструктор
        3. експлицитен конструктор
    */
    CFlowers(){
        flowerName = "Default flower name";
        isFlowering = false;
        flowerFamily = "Default flower family";
        flowerType = annual;
    }
    CFlowers(const CFlowers& fl){
        flowerName = fl.flowerName;
        isFlowering = fl.isFlowering;
        flowerFamily = fl.flowerFamily;
        flowerType = fl.flowerType;
    }
    CFlowers(const string& name, bool isF, const string& fam, const FlowerType&
type):
        flowerName(name), isFlowering (isF), flowerFamily (fam),
flowerType(type){}
    CFlowers(istream& flowerInfo):CFlowers(){
        string line;
        getline(flowerInfo,line);
        if(isNameValid(line)){
            string name = line;
            getline(flowerInfo,line);
            if(isFloweringValid(line)){
                bool isF;
                line=="цъфти"?isF=true:isF=false;
                getline(flowerInfo,line);
                if(isFamilyValid(line)){
```

```
                    string fam = line;
                    getline(flowerInfo,line);
                    if(isTypeValid(line)){
                        FlowerType type;
                        if(line=="едногодишно")
                            type = annual;
                        if(line=="двугодишно")
                            type = biennial;
                        if(line=="многогодишно")
                            type = perennial;
                        *this = CFlowers(name,isF,fam,type);
                    }
                }
            }
        }
    }
    /*
    Съставете функции за:
        1. за установяване на член променливите
        2. функция за извеждане в изходен поток (отделно файл, отделно конзола)
    */
    void setFlowerName(const string& name){
        flowerName = name;
    }
    void setIsFlowering(bool isF){
        isFlowering = isF;
    }
    void setFlowerFamily(const string& fam){
        flowerName = fam;
    }
    void setFlowerType(const FlowerType& type){
        flowerType = type;
    }
    void consoleOutput(){
        cout<<*this;
    }
    void fileOutput(const string& filePath){
        try{
            ofstream oFile;
            oFile.open(filePath, ios::out);
            if(!oFile.is_open())
                throw -1;
            oFile<<*this;
            oFile.close();
        }
        catch (int iEx){
            cout<<"Something went wrong while opening file\n";
        }
    }
    /*
    Съставете функции за:
        1. Защита от невалидни данни
        2. Определя дали в името на цветето се съдържа словосъчетание от името
на
        семейството (пр. папрат папратови)
        3. Търсене по семейство
        4. Оператор за присвояване =
        5. Логически оператор, който проверява дали цветето е цъфтящо
        6. Оператор за съвпадение (по семейство).
    */
    bool isNameValid(const string& flowerName){
        return flowerName!="";
    }
    bool isFloweringValid(const string& isFlowering){
```

```
        if(isFlowering=="цъфти"||isFlowering=="не цъфти")
            return true;
        return false;
    }
    bool isFamilyValid(const string& flowerFamily){
        return flowerFamily!="";
    }
    bool isTypeValid(const string& flowerType){
        if(flowerType=="едногодишно"||flowerType=="двугодишно"||
flowerType=="многогодишно")
            return true;
        return false;
    }
    bool isFamilyInName(){
        string toFind;
        if(flowerFamily.size()<8)
            toFind = flowerFamily;
        else
            toFind = flowerFamily.substr(0,8);
        if(flowerName.find(toFind)==flowerName.npos)
            return false;
        return true;
    }
    bool sameFamily(const string& fam){
        return flowerFamily==fam;
    }
    void operator=(const CFlowers& fl){
        flowerName = fl.flowerName;
        isFlowering = fl.isFlowering;
        flowerFamily = fl.flowerFamily;
        flowerType = fl.flowerType;
    }
    bool operator>(const CFlowers& dummy){
        return isFlowering;
    }
    bool operator==(const CFlowers& fl){
        return flowerFamily==fl.flowerFamily;
    }
    string getFlowerName(){
        return flowerName;
    }
    string getFlowerFamily(){
        return flowerFamily;
    }
    FlowerType getFlowerType(){
        return flowerType;
    }
    friend ostream& operator<<(ostream& toStream, const CFlowers& fl){
        toStream<<fl.flowerName<<endl<<(fl.isFlowering?"цъфти\n":"не цъфти\
n")<<fl.flowerFamily<<endl;
        switch (fl.flowerType)
        {
        case annual:
            toStream<<"едногодишно\n";
            break;
        case biennial:
            toStream<<"двугодишно\n";
            break;
        default:
            toStream<<"многогодишно\n";
        }
        return toStream;
    }
};
```

```
/*
II. Да се състави клас CParnik, който съдържа CFlowers с член променливи име на
парника, брой цветя, брой приходи за една година.
*/
class CParnik{
    vector<CFlowers> flowersInside;
    string parnikName;
    int flowerCount;
    int yearlyIncome;
    public:
    /*
    Съставете следните функции:
        1. Създаване на обект чрез друг обект
        2. Създаване на обект чрез експлицитно зададени параметри
        3. Функция, определяща броя цветя дали надвишава 1000.
    */
    CParnik(){
        flowersInside = vector<CFlowers>(1);
        parnikName = "Default parnik";
        flowerCount = flowersInside.size();
        yearlyIncome = 0;
    }
    CParnik(const CParnik& p){
        flowersInside = p.flowersInside;
        parnikName = p.parnikName;
        flowerCount = p.flowerCount;
        yearlyIncome = p.yearlyIncome;
    }
    CParnik(const string& pn, int fc, int yi){
        flowersInside = vector<CFlowers>(fc);
        parnikName = pn;
        flowerCount = fc;
        yearlyIncome = yi;
    }
    CParnik(const string& filePath){
        try{
            ifstream iFile(filePath);
            if(!iFile.is_open()) throw -1;
            string line;
            getline(iFile,line);
            parnikName = line;
            getline(iFile,line);
            flowerCount = stoi(line);
            getline(iFile,line);
            yearlyIncome = stoi(line);
            for(int i = 0; i< flowerCount; i++){
                stringstream flower;
                for(int j = 0; j < 4; j++){
                    getline(iFile,line);
                    flower<<line<<endl;
                }
                flowersInside.push_back(CFlowers(flower));
            }
        } catch (...) {
            cout<<"Something went wrong while loading file: "+filePath<<endl;
            *this = CParnik();
        }
    }
    bool areFlowersMoreThanA1000(){
        return flowerCount>1000;
    }
    void setParnikName(const string& n){
        parnikName = n;
```

```
    }
    bool allFlowering(){
        CFlowers dummy;
        for(auto it = flowersInside.begin(); it!= flowersInside.end(); it++)
            if((*it)>dummy)
                return true;
        return false;
    }
    int getYearlyIncome(){
        return yearlyIncome;
    }
    int getFlowerCount(){
        return flowerCount;
    }
    vector<CFlowers>& getFlowersInside(){
        return flowersInside;
    }
    bool operator<(const CParnik& p){
        return yearlyIncome<p.yearlyIncome;
    }
    bool operator==(const CParnik p){
        return yearlyIncome==p.yearlyIncome;
    }
    friend ostream& operator<<(ostream& toStream, const CParnik& p){
        toStream<<p.parnikName<<endl
        <<p.flowerCount<<endl
        <<p.yearlyIncome<<endl;
        for(auto it = p.flowersInside.begin(); it!= p.flowersInside.end();it++)
            toStream<<*it;

        return toStream;
    }
};

/*
III. Съставете главна програма, илюстрираща използването на създадените класове.
Да
се създаде контейнер вектор от 10 обекта от класа CParnik и да се направят
следните
справки, като:
    1. извеждане в отделен списък (list) на цветята едногодишни, срещащи се най-
    малко в 3 парника
    2. Функция, определяща от всяко семейство колко е броя на цветята.
    3. да се изведат в отделни файлове всички парници, които отглеждат само
    цъфтящи с брой приходи над 100000 лева на година
    4. да се изведат цветята, които са над 20 за години, чийто тип е подаден
като
    параметър
    5. да се изведат всички данни за парниците, с брой приходи, получени като
    параметър от текстов файл
    6. при подаден като параметър общ брой цветя, да се изведат парниците, които
    са с приходи от първата тройка в отделен контейнер.
*/
class MainClass{
    vector<CParnik> ps;
    public:
    MainClass(){
        ps.reserve(10);
        ps.push_back(CParnik());
        ps[0].setParnikName("Парник 1");
        ps.push_back(CParnik("Парник 2",3,1000));
        for(int i = 3; i < 10; i++){
            ps.push_back(CParnik(("parnici/p"+to_string(i)+".txt")));
        }
```

```
    ps.push_back(CParnik(ps[8]));
    ps[9].setParnikName("Парник 10");
    //for(auto it = ps.begin(); it!= ps.end(); it++)
        //cout<<*it;
}
list<CFlowers> getFlowersInAtLeast3P(){
    vector<CFlowers> cfls;
    vector<string> flNames;
    vector<int> counts;
    vector<int> lastIt;
    string fl;
    int currIteration = 1;
    for(auto it = ps.begin(); it!= ps.end(); it++){
        vector<CFlowers> flRef = (*it).getFlowersInside();
        for(auto it2 = flRef.begin(); it2!= flRef.end(); it2++){
            fl = (*it2).getFlowerName();
            auto it3 = find(flNames.begin(),flNames.end(),fl);
            if(it3==flNames.end()){
                cfls.push_back(*it2);
                flNames.push_back(fl);
                counts.push_back(1);
                lastIt.push_back(currIteration);
            } else if(lastIt[distance(flNames.begin(),it3)]==currIteration){
                continue;
            } else {
                counts[distance(flNames.begin(),it3)]++;
                lastIt[distance(flNames.begin(),it3)]=currIteration;
            }
        }
        currIteration++;
    }
    list<CFlowers> toRet;
    int size = cfls.size();
    for(int i=0; i<size; i++)
        if(counts[i]>=3)
            toRet.push_back(cfls[i]);
    return toRet;
}
void outputFamiliesWithCount(){
    vector<string> families;
    vector<int> counts;
    string fam;
    for(auto it = ps.begin(); it!= ps.end(); it++){
        vector<CFlowers> flRef = (*it).getFlowersInside();
        for(auto it2 = flRef.begin(); it2!= flRef.end(); it2++){
            fam = (*it2).getFlowerFamily();
            auto it3 = find(families.begin(),families.end(),fam);
            if(it3==families.end()){
                families.push_back(fam);
                counts.push_back(1);
            } else {
                counts[distance(families.begin(),it3)]++;
            }
        }
    }
    int size = families.size();
    for(int i=0; i<size; i++){
        cout<<families[i]<<' '<<counts[i]<<endl;
    }
}
void outputFloweringParniciOver100000(){
    sort(ps.rbegin(),ps.rend());
    int counter = 1;
    for(auto it = ps.begin(); it!= ps.end(); it++, counter++){
```

```
            if((*it).getYearlyIncome()>100000){
                if((*it).allFlowering()){
                    try{
                        ofstream oFile;
                        oFile.open(("p_outputs/p_out"+to_string(counter)
+".txt"), ios::out|ios::trunc);
                        if(!oFile.is_open()) throw -1;
                        oFile<<*it;
                        oFile.close();
                    }
                    catch (int iEx){
                        cout<<"Something went wrong while opening file\n";
                    }
                }
            } else break;
        }
    }
    void outputFlowersOver20ByType(const FlowerType& t){
        vector<CFlowers> cfls;
        vector<string> flNames;
        vector<int> counts;
        string fl;
        for(auto it = ps.begin(); it!= ps.end(); it++){
            vector<CFlowers> flRef = (*it).getFlowersInside();
            for(auto it2 = flRef.begin(); it2!= flRef.end(); it2++){
                if((*it2).getFlowerType()!=t) continue;
                fl = (*it2).getFlowerName();
                auto it3 = find(flNames.begin(),flNames.end(),fl);
                if(it3==flNames.end()){
                    cfls.push_back(*it2);
                    flNames.push_back(fl);
                    counts.push_back(1);
                } else {
                    counts[distance(flNames.begin(),it3)]++;
                }
            }
        }
        int size = cfls.size();
        for(int i=0; i<size; i++){
            if(counts[i]>20)
                cout<<cfls[i]<<"of which there are: "<<counts[i]<<endl;
        }
    }
    void outputByIncomeFromFile(const string& filePath){
        try{
            ifstream iFile(filePath);
            string line;
            int income;
            getline(iFile, line);
            income = stoi(line);
            for(auto it = ps.begin(); it!= ps.end(); it++)
                if((*it).getYearlyIncome()==income)
                    cout<<*it;
        } catch (...) {
            cout<<"Error in file"<<endl;
        }
    }
    vector<CParnik> getRichestParnici(int& flowersInAll3){
        vector<CParnik> toRet;
        toRet.reserve(3);
        sort(ps.rbegin(),ps.rend());
        flowersInAll3 = 0;
        auto it = ps.begin();
        for(int i = 0; i < 3 ; i++, it++){
```

```
            flowersInAll3+=(*it).getFlowerCount();
            toRet.push_back(*it);
        }
        return toRet;
    }

};

int main (){
    MainClass m;

    //1. извеждане в отделен списък (list) на цветята едногодишни, срещащи се
най-
    //малко в 3 парника
    cout<<"These flowers are in at least 3 parnici:"<<endl;
    list<CFlowers> l = m.getFlowersInAtLeast3P();
    for(auto it = l.begin(); it!= l.end(); it++)
        cout<<*it;
    cout<<endl;

    //2. Функция, определяща от всяко семейство колко е броя на цветята.
    cout<<"The following flower families are in the parnici:"<<endl;
    m.outputFamiliesWithCount();
    cout<<endl;

    //3. да се изведат в отделни файлове всички парници, които отглеждат само
    //цъфтящи с брой приходи над 100000 лева на година
    m.outputFloweringParniciOver100000();

    //4. да се изведат цветята, които са над 20 за години, чийто тип е подаден
като
    //параметър
    cout<<"The following flowers are over 20 and the given type:"<<endl;
    cout<<"annual:"<<endl;
    m.outputFlowersOver20ByType(annual);
    cout<<"biennial:"<<endl;
    m.outputFlowersOver20ByType(biennial);
    cout<<"perennial:"<<endl;
    m.outputFlowersOver20ByType(perennial);
    cout<<endl;

    //5. да се изведат всички данни за парниците, с брой приходи, получени като
    //параметър от текстов файл
    cout<<"Parnici with income given by file:"<<endl;
    m.outputByIncomeFromFile("income.txt");

    //6. при подаден като параметър общ брой цветя, да се изведат парниците,
които
    //са с приходи от първата тройка в отделен контейнер.
    int flowersInBest;
    vector<CParnik> bestPs = m.getRichestParnici(flowersInBest);
    cout<<endl<<"There are "<<flowersInBest<<" flowers in the best
parnici."<<endl;
    cout<<"Which are:"<<endl;
    for(auto it = bestPs.begin(); it!= bestPs.end(); it++)
        cout<<*it;

    return 0;
}
```

Конзолен изход от задачата:

```
These flowers are in at
least 3 parnici:
Агератум
цъфти
Астрови
едногодишно
Азарина виолетова
цъфти
Живовлекови
едногодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно

The following flower
families are in the
parnici:
Default flower family 4
Астрови 31
Живовлекови 31
Щирови 3
Божурови 22
Кръстоцветни 5
Сложноцветни 1
Сенникоцветни 2
Камбанкови 2
Лъжичничета 2

The following flowers are
over 20 and the given
type:
annual:
Агератум
цъфти
Астрови
едногодишно
of which there are: 27
biennial:
Пъстър напръстник
цъфти
Живовлекови
двугодишно
of which there are: 23
perennial:
Божур
цъфти
Божурови
многогодишно
of which there are: 22

Parnici with income given
by file:
Парник 2
3
1000
Default flower name
не цъфти
Default flower family
едногодишно
Default flower name
не цъфти
Default flower family
едногодишно
Парник 3
7
1000
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Азарина виолетова
цъфти
Живовлекови
едногодишно
Азарина виолетова
цъфти
Живовлекови
едногодишно
Азарина виолетова
цъфти
Живовлекови
едногодишно
Азарина виолетова
цъфти
Живовлекови
едногодишно
Парник 4
8
1000
Агератум
цъфти
Астрови
едногодишно
Азарина виолетова
цъфти
Живовлекови
едногодишно
Амарантус ПЕРФЕКТА
цъфти
Щирови
едногодишно
Амарантус ПЕРФЕКТА
цъфти
Щирови
едногодишно
Астри Божуровидни
цъфти
Астрови
едногодишно
Астри Божуровидни
цъфти
Астрови
едногодишно
Астри Божуровидни
цъфти
Астрови
едногодишно
Астри Божуровидни
цъфти
Астрови
едногодишно
Парник 4
9
1000
Агератум
цъфти
Астрови
едногодишно
Азарина виолетова
цъфти
Живовлекови
едногодишно
Амарантус ПЕРФЕКТА
цъфти
Щирови
едногодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Аубриета ПУРПУРНА КАСКАДА
цъфти
Кръстоцветни
многогодишно
Аубриета ПУРПУРНА КАСКАДА
цъфти
Кръстоцветни
многогодишно
Аубриета ПУРПУРНА КАСКАДА
цъфти
Кръстоцветни
многогодишно

There are 61 flowers in
the best parnici.
Which are:
Парник 6
20
100001
```

Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно

Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Парник 7
21
100001
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник

цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Парник 8
20
100001
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти

```
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Метличина
цъфти
Сложноцветни
едногодишно
```

Файлов изход от задачата:

- *във файл* p_out1.txt:

```
Парник 6
20
100001
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
Агератум
цъфти
Астрови
едногодишно
```

- *във файл* p_out2.txt:

```
Парник 7
21
100001
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
Пъстър напръстник
цъфти
Живовлекови
двугодишно
```

- *във файл* p_out3.txt:

```
Парник 8
20
100001
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Божур
цъфти
Божурови
многогодишно
Метличина
цъфти
Сложноцветни
едногодишно
```