**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИТМО»**

**ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**ОТЧЕТ**

**по Лабораторной работе № 20**

**«Введение в понятие класса на языке C++»**

Специальность 09.02.07«Информационные системы и программирование»

Дисциплина «Основы алгоритмов и программирование»

|  |  |
| --- | --- |
| Преподаватель:  Яковлева Н.М.  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_202\_\_г.  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Выполнил:  студент группы Y2231  Петров Н.В. |

Санкт-Петербург

2019/2020

**Задание**

1. Создать класс “Студент”, включающий следующие элементы:

поля:

* ФИО (задаётся динамически), дата рождения, оценки за три последние лабораторные работы по ОАиП, наличие или отсутствие стипендии
* статическое поле количества студентов
* явный конструктор
* явный деструктор (так как используется динамическая память)
* метод вывода на экран всех полей
* метод вычисления средней оценки, на основе которой меняется поле наличия или отсутствия стипендии
* статический метод изменения количества студентов при создании или удалении студента
* перегруженные операции копирования и присваивания
* перегруженную операцию > сравнения положения в алфавите для ФИО

1. Также создать дружественные классу “Студент” функции:

* выводящую студентов, у которых есть стипендия
* сортирующую студентов по ФИО (не просто выводит упорядоченно, а сортирует созданный массив (или иную структуру данных) из студентов

1. Ваша программа должна содержать меню, позволяющее выбрать действие: например, создать студента, осуществить сортировку и т.д.

**Исходный код**

main.cpp

#include <iostream>  
#include <cstring>  
  
using namespace std;  
  
#define loop while (true)  
#define VEC\_BLOCK\_SIZE 10  
#define VEC\_TYPE Student  
#define BUFF\_SIZE 256  
  
class Student {  
 char \*surname;  
 char \*name;  
 char \*patronymic;  
 short int marks[3];  
 char \*dob;  
 bool grants;  
public:  
 static int quantity;  
  
 Student() {  
 this->surname = nullptr;  
 this->name = nullptr;  
 this->patronymic = nullptr;  
 \*this->marks = 0;  
 this->dob = nullptr;  
 this->grants = false;  
 }  
  
 Student(char \*surname, char \*name, char \*patronymic, short int marks[3], char \*dob) {  
 this->surname = new char[strlen(surname) + 1];  
 this->name = new char[strlen(name) + 1];  
 this->patronymic = new char[strlen(patronymic) + 1];  
 strcpy(this->surname, surname);  
 strcpy(this->name, name);  
 strcpy(this->patronymic, patronymic);  
 memcpy(this->marks, marks, sizeof(short int) \* 3);  
 this->dob = new char[strlen(dob) + 1];  
 strcpy(this->dob, dob);  
 Student::quantity++;  
 this->grants = this->recount();  
 }  
  
 ~Student() {  
 if (this->name == nullptr) return;  
 delete this->surname;  
 delete this->name;  
 delete this->patronymic;  
 delete this->dob;  
 this->name = nullptr;  
 }  
  
 Student &operator=(const Student &t) {  
 if (this == &t) return \*this;  
 this->surname = t.surname;  
 this->name = t.name;  
 this->patronymic = t.patronymic;  
 \*this->marks = \*t.marks;  
 this->dob = t.dob;  
 this->grants = t.grants;  
 return \*this;  
 }  
  
 Student copy() {  
 Student n;  
 n.surname = new char[strlen(this->surname) + 1];  
 n.name = new char[strlen(this->name) + 1];  
 n.patronymic = new char[strlen(this->patronymic) + 1];  
 strcpy(n.surname, this->surname);  
 strcpy(n.name, this->name);  
 strcpy(n.patronymic, this->patronymic);  
 memcpy(n.marks, this->marks, sizeof(short int) \* 3);  
 n.dob = new char[strlen(this->dob) + 1];  
 strcpy(n.dob, this->dob);  
 n.grants = this->grants;  
 return n;  
 }  
  
 bool operator>(const Student &s) {  
 if (this == &s) return false;  
 char \*name\_1, \*name\_2;  
 unsigned long size\_1 = strlen(s.surname) + strlen(s.name) + strlen(s.patronymic) - 2;  
 unsigned long size\_2 = strlen(this->surname) + strlen(this->name) + strlen(this->patronymic) - 2;  
 name\_1 = new char[size\_1];  
 name\_2 = new char[size\_2];  
 strcpy(name\_1, s.surname);  
 strcpy(strchr(name\_1, '\0'), s.name);  
 strcpy(strchr(name\_1, '\0'), s.patronymic);  
 strcpy(name\_2, this->surname);  
 strcpy(strchr(name\_2, '\0'), this->name);  
 strcpy(strchr(name\_2, '\0'), this->patronymic);  
 bool out = false;  
 for (int i = 0; i < size\_1 < size\_2 ? size\_1 : size\_2; i++) {  
 if (name\_1[i] > name\_2[i]) {  
 out = true;  
 break;  
 }  
 }  
 return out;  
 }  
  
 bool recount() {  
 return this->grants = (marks[0] + marks[1] + marks[2]) / 3 > 4;  
 }  
  
 void print() {  
 cout << "ФИО: " << this->surname << " " << this->name << " " << this->patronymic << endl;  
 cout << "Оценки: " << this->marks[0] << " " << this->marks[1] << " " << this->marks[2] << " " << endl;  
 cout << "Наличие стипендии: ";  
 if (this->grants) cout << "+";  
 else cout << "-";  
 cout << endl;  
 cout << "Дата рождения: " << this->dob << endl;  
 }  
};  
  
int Student::quantity = 0;  
  
class Vector {  
 VEC\_TYPE \*\*vec;  
 int size;  
 int capacity;  
public:  
 Vector() {  
 vec = new class VEC\_TYPE \*[VEC\_BLOCK\_SIZE];  
 size = 0;  
 capacity = VEC\_BLOCK\_SIZE;  
 }  
  
 ~Vector() {  
 delete[] this->vec;  
 }  
  
 int len() {  
 return this->size;  
 }  
  
 void append(VEC\_TYPE \*element) {  
 if (size >= capacity) {  
 VEC\_TYPE \*\*tmp = new class VEC\_TYPE \*[this->size];  
 memcpy(tmp, this->vec, sizeof(VEC\_TYPE \*) \* this->size);  
 this->vec = new class VEC\_TYPE \*[this->size + VEC\_BLOCK\_SIZE];  
 memcpy(this->vec, tmp, sizeof(VEC\_TYPE \*) \* this->size);  
 delete[] tmp;  
 capacity += VEC\_BLOCK\_SIZE;  
 }  
 this->vec[size++] = element;  
 }  
  
 VEC\_TYPE \*get(int index) {  
 return this->vec[index];  
 }  
  
 void pop() {  
 if (size == 0) {  
 cout << "Нельзя удалить ничего!" << endl;  
 return;  
 }  
 if ((capacity - size) % VEC\_BLOCK\_SIZE == 0) {  
 VEC\_TYPE \*\*tmp = new class VEC\_TYPE \*[this->size - VEC\_BLOCK\_SIZE];  
 memcpy(tmp, this->vec, sizeof(VEC\_TYPE \*) \* this->size - VEC\_BLOCK\_SIZE);  
 delete[] this->vec;  
 this->vec = tmp;  
 capacity -= VEC\_BLOCK\_SIZE;  
 }  
 this->vec[--size] = nullptr;  
 }  
  
 int search(Student \*element) {  
 for (int i = 0; i < this->size; i++) {  
 if (vec[i] == element) {  
 return i;  
 }  
 }  
 return -1;  
 }  
  
 void sort() {  
 bool is\_sorted = false;  
 while (!is\_sorted) {  
 is\_sorted = true;  
 for (int i = 0; i < this->size - 1; i++) {  
 if (this->vec[i] < this->vec[i + 1]) {  
 is\_sorted = false;  
 VEC\_TYPE \*tmp = this->vec[i];  
 this->vec[i] = this->vec[i + 1];  
 this->vec[i + 1] = tmp;  
 }  
 }  
 }  
 }  
};  
  
struct Buffer {  
 char surname[BUFF\_SIZE];  
 char name[BUFF\_SIZE];  
 char patronymic[BUFF\_SIZE];  
 short int marks[3];  
 char dob[BUFF\_SIZE];  
};  
  
int main() {  
 Vector vec;  
 char mode;  
 loop {  
 cout << "1. Ввести нового студента\n"  
 "2. Вывести всех студентов\n"  
 "3. Удалить последнего студента\n"  
 "4. Сортировать список студентов\n"  
 "5. Выйти из программы\n"  
 "Выберете режим работы: ";  
 cin >> mode;  
 switch (mode) {  
 case '1': {  
 Buffer buff{};  
 cout << "Введите фамилию: " << endl;  
 cin >> buff.surname;  
 cout << "Введите имя: " << endl;  
 cin >> buff.name;  
 cout << "Введите отчество: " << endl;  
 cin >> buff.patronymic;  
 cout << "Введите дату рождения в формате DD.MM.YYYY: " << endl;  
 cin >> buff.dob;  
 cout << "Введите первую оценку: " << endl;  
 cin >> buff.marks[0];  
 cout << "Введите вторую оценку: " << endl;  
 cin >> buff.marks[1];  
 cout << "Введите третью оценку: " << endl;  
 cin >> buff.marks[2];  
 auto \*tmp = new Student(buff.surname, buff.name, buff.patronymic, buff.marks, buff.dob);  
 vec.append(tmp);  
 break;  
 }  
 case '2':  
 for (int i = 0; i < vec.len(); i++) vec.get(i)->print();  
 break;  
 case '3':  
 vec.pop();  
 break;  
 case '4':  
 vec.sort();  
 break;  
 case '5':  
 return 0;  
 default:  
 cout << "Это не 1, не 2, не 3, не 4 и даже не 5" << endl;  
 }  
  
 }  
}

**Результат выполнения**

1. Ввести нового студента  
2. Вывести всех студентов  
3. Удалить последнего студента  
4. Сортировать список студентов  
5. Выйти из программы  
Выберете режим работы: 1  
Введите фамилию:  
Головачев  
Введите имя:  
Данила  
Введите отчество:  
Андреевич  
Введите дату рождения в формате DD.MM.YYYY:  
12.04.2001  
Введите первую оценку:  
5  
Введите вторую оценку:  
5  
Введите третью оценку:  
5  
1. Ввести нового студента  
2. Вывести всех студентов  
3. Удалить последнего студента  
4. Сортировать список студентов  
5. Выйти из программы  
Выберете режим работы: 1  
Введите фамилию:  
Аркадьев  
Введите имя:  
Аркадий  
Введите отчество:  
Аркадьевич  
Введите дату рождения в формате DD.MM.YYYY:  
31.06.2000  
Введите первую оценку:  
3  
Введите вторую оценку:  
3  
Введите третью оценку:  
4  
1. Ввести нового студента  
2. Вывести всех студентов  
3. Удалить последнего студента  
4. Сортировать список студентов  
5. Выйти из программы  
Выберете режим работы: 2  
ФИО: Головачев Данила Андреевич  
Оценки: 5 5 5  
Наличие стипендии: +  
Дата рождения: 12.04.2001  
ФИО: Аркадьев Аркадий Аркадьевич  
Оценки: 3 3 4  
Наличие стипендии: -  
Дата рождения: 31.06.2000  
1. Ввести нового студента  
2. Вывести всех студентов  
3. Удалить последнего студента  
4. Сортировать список студентов  
5. Выйти из программы  
Выберете режим работы: 4  
1. Ввести нового студента  
2. Вывести всех студентов  
3. Удалить последнего студента  
4. Сортировать список студентов  
5. Выйти из программы  
Выберете режим работы: 2  
ФИО: Аркадьев Аркадий Аркадьевич  
Оценки: 3 3 4  
Наличие стипендии: -  
Дата рождения: 31.06.2000  
ФИО: Головачев Данила Андреевич  
Оценки: 5 5 5  
Наличие стипендии: +  
Дата рождения: 12.04.2001  
1. Ввести нового студента  
2. Вывести всех студентов  
3. Удалить последнего студента  
4. Сортировать список студентов  
5. Выйти из программы  
Выберете режим работы: 3  
1. Ввести нового студента  
2. Вывести всех студентов  
3. Удалить последнего студента  
4. Сортировать список студентов  
5. Выйти из программы  
Выберете режим работы: 2  
ФИО: Аркадьев Аркадий Аркадьевич  
Оценки: 3 3 4  
Наличие стипендии: -  
Дата рождения: 31.06.2000  
1. Ввести нового студента  
2. Вывести всех студентов  
3. Удалить последнего студента  
4. Сортировать список студентов  
5. Выйти из программы  
Выберете режим работы: 5