Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №3**

По дисциплине: “Языки программирования”

Тема: “Перегрузка операций. Исключения”**Вариант №3**

**Выполнил**: студент 2 курса группы ПО-7 Панкратов Роман Сергеевич

**Проверила:** Бойко Д.О.

Брест 2021

### Цель работы

Изучение правил перегрузки операций и принципов обработки исключений в C++

### Общие требования

В начале программы вывести задание; в процессе работы выводить подсказки пользователю (что ему нужно ввести, чтобы продолжить выполнение программы). Иерархию классов следует взять из лабораторной работы №3. Класс коллекция может не иметь методов для изменения количества хранимых объектов. При обращении к элементам с несуществующим индексом должно выбрасываться исключение. После работы программы вся динамически выделенная память должна быть освобождена.

### Задание

**3.** Написать программу, в которой описана иерархия классов: человек (дошкольник, школьник, студент, рабочий). Описать класс для хранения коллекции людей (массива указателей на базовый класс), в котором перегрузить операцию «[ ]». Для базового класса и его потомков перегрузить операции «==», «!=», «=». Продемонстрировать работу операторов.

# Код программы

main.cpp

#include **<iostream>**#include **"classes.h"  
  
int** main() {  
 **auto** pre\_schooler = **new** PreSchooler(**"Roman"**);  
 **auto** school\_boy = **new** SchoolBoy(**"Igor"**, 15);  
 **auto** student = **new** Student(**"Vanya"**, 19, 2);  
 **auto** worker = **new** Worker(**"Dima"**, 22, 4, 1500.0);  
  
 **auto** array = **new** PeopleArray();  
 array->append(pre\_schooler);  
 array->append(school\_boy);  
 array->append(student);  
 array->append(worker);  
  
 **for** (**int** i = 0; i < array->len(); i++) {  
 (\*array)[i]->print();  
 }  
  
 \*school\_boy = \*student;  
 std::cout << std::endl;  
  
 **for** (**int** i = 0; i < array->len(); i++) {  
 (\*array)[i]->print();  
 }  
  
 std::cout << std::endl;  
  
 **bool** res2 = \*pre\_schooler == \*school\_boy;  
 **bool** res3 = \*pre\_schooler != \*worker;  
  
 std::cout << res2 << std::endl;  
 std::cout << res3 << std::endl;

**delete** pre\_schooler;  
**delete** school\_boy;  
**delete** student;  
**delete** worker;

**return** 0;  
}

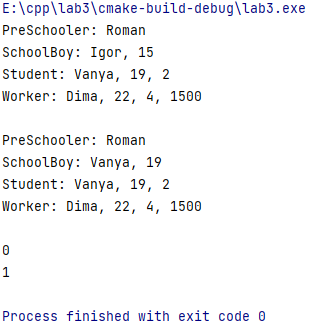
classes.h

#include **"string"  
  
class** PreSchooler {  
**protected**:  
 std::string \_name;  
  
**public**:  
 PreSchooler(std::string name);  
  
 **bool operator**== (PreSchooler &p);  
 **bool operator**!= (PreSchooler &p);  
 PreSchooler& **operator**= (PreSchooler &p);  
  
 **virtual void** print();  
};  
  
**class** SchoolBoy: **public** PreSchooler {  
**protected**:  
 **int** \_age;  
  
**public**:  
 SchoolBoy(std::string name, **int** age);  
  
 **void** print() **override**;  
};  
  
**class** Student: **public** SchoolBoy {  
**protected**:  
 **int** \_course;  
  
**public**:  
 Student(std::string name, **int** age, **int** course);  
  
 **void** print() **override**;  
};  
  
**class** Worker: **public** Student {  
**protected**:  
 **float** \_salary;  
  
**public**:  
 Worker(std::string name, **int** age, **int** course, **float** salary);  
  
 **void** print() **override**;  
};  
  
**class** PeopleArray{  
 **int** length = 0;  
 PreSchooler\*\* array = **new** PreSchooler\*[length];  
  
**public**:  
 **int** len();  
 **void** append(PreSchooler\* p);  
  
 PreSchooler\* **operator**[](**int** index);  
};

classes.cpp

#include **"classes.h"**#include **<iostream>**PreSchooler::PreSchooler(std::string name) :\_name(name) {}  
  
**bool** PreSchooler::**operator**==(PreSchooler &p) {  
 **return** \_name == p.\_name;  
}  
  
**void** PreSchooler::print() {  
 std::cout << **"PreSchooler: "** << \_name << std::endl;  
}  
  
**bool** PreSchooler::**operator**!=(PreSchooler &p) {  
 **return** \_name != p.\_name;  
}  
  
PreSchooler& PreSchooler::**operator**=(PreSchooler &p) {  
 \_name = p.\_name;  
 **return** \***this**;  
}  
  
SchoolBoy::SchoolBoy(std::string name, **int** age): PreSchooler(name), \_age(age) {}  
  
**void** SchoolBoy::print() {  
 std::cout << **"SchoolBoy: "** << \_name << **", "** << \_age << std::endl;  
}  
  
Student::Student(std::string name, **int** age, **int** course): SchoolBoy(name, age), \_course(course) {}  
  
**void** Student::print() {  
 std::cout << **"Student: "** << \_name << **", "** << \_age << **", "** << \_course << std::endl;  
}  
  
Worker::Worker(std::string name, **int** age, **int** course, **float** salary): Student(name, age, course), \_salary(salary) {}  
  
**void** Worker::print() {  
 std::cout << **"Worker: "** << \_name << **", "** << \_age << **", "** << \_course << **", "** << \_salary << std::endl;  
}  
  
**int** PeopleArray::len() {  
 **return** length;  
}  
  
**void** PeopleArray::append(PreSchooler \*p) {  
 **auto**\*\* new\_array = **new** PreSchooler\*[length+1];  
 **for** (**int** i = 0; i < length; i++) {  
 new\_array[i] = array[i];  
 }  
 new\_array[length] = p;  
 array=new\_array;  
 length += 1;  
}  
  
PreSchooler \*PeopleArray::**operator**[](**int** index) {  
 **return** array[index];  
}

# Результаты программы



### Вывод

Я изучил правила перегрузки операций и принципов обработки исключений в C++