Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема : "Простое наследование. Принцип подстановки."

Выполнила работу

Студентка группы РИС-22-1Б

Верхоланцева Е. С.

Проверила

Доцент кафедры ИТАС

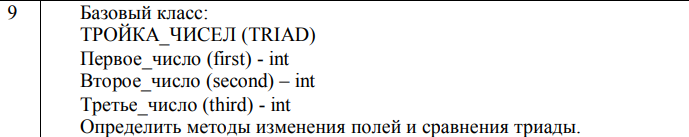
Полякова О.А.

Г. Пермь-2023

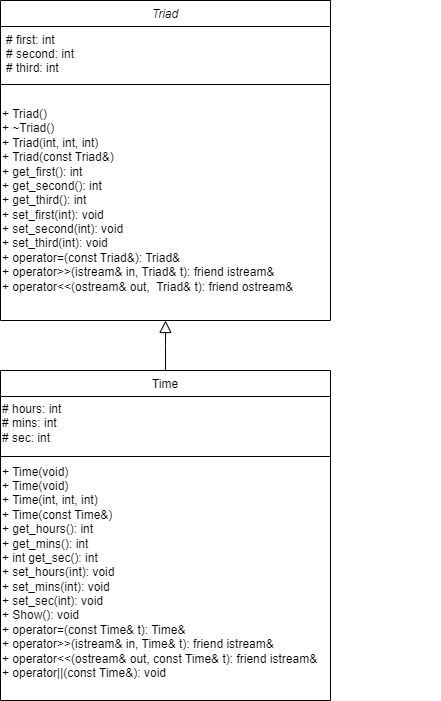
**Постановка задачи**

1. Определить пользовательский класс.  
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.  
3. Определить в классе деструктор.  
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).  
5. Перегрузить операцию присваивания.  
6. Перегрузить операции ввода и вывода объектов с помощью потоков.  
7. Определить производный класс.  
8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.  
9. Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.

Вариант 9



**Диаграмма класса**



**Описание компонентных функций**

**triad.cpp**

Triad::Triad()

{

first = 0;

second = 0;

third = 0;

cout << "Вызван конструктор без параметров для объекта " << this << endl;

}

Triad::~Triad()

{

cout << "Вызван деструктор для объекта " << this << endl;

}

Triad::Triad(int valueF, int valueS, int valueT)

{

first = valueF;

second = valueS;

third = valueT;

}

Triad::Triad(const Triad& triad)

{

first = triad.first;

second = triad.second;

third = triad.third;

}

void Triad::set\_first(int f) { first = f; }

void Triad::set\_second(int s) { second = s; }

void Triad::set\_third(int t) { third = t; }

int Triad::get\_first() { return first; }

int Triad::get\_second() { return second; }

int Triad::get\_third() { return third; }

Triad& Triad::operator=(const Triad& t)

{

if (&t == this) return \*this;

first = t.first;

second = t.second;

third = t.third;

return \*this;

}

istream& operator>>(istream& in, Triad& t)

{

cout << "Первое число: "; in >> t.first;

cout << "Второе число: "; in >> t.second;

cout << "Третье число: "; in >> t.third;

cout << "\n";

return in;

}

ostream& operator<<(ostream& out, Triad& t)

{

out << "\nПервое число: " << t.get\_first();

out << "\nВторое число: " << t.get\_second();

out << "\nТретье число: " << t.get\_third();

return out;

}

**Time.cpp**

Time::Time(void) :Triad() {

hours = 0;

mins = 0;

sec = 0;

}

Time::~Time(void) {}

Time::Time(int h, int m, int s) :Triad(first, second, third)

{

hours = h;

mins = m;

sec = s;

}

Time::Time(const Time& t)

{

first = t.first;

second = t.second;

third = t.third;

hours = t.hours;

mins = t.mins;

sec = t.sec;

}

// селекторы

int Time::get\_hours() { return hours; }

int Time::get\_mins() { return mins; }

int Time::get\_sec() { return sec; }

// модификаторы

void Time::set\_hours(int h) { hours = h; }

void Time::set\_mins(int m) { mins = m; }

void Time::set\_sec(int s) { sec = s; }

// перегрузка оператора присваивания

Time& Time::operator=(const Time& t)

{

if (&t == this) return \*this;

first = t.first;

second = t.second;

third = t.third;

Time::hours = t.hours;

Time::mins = t.mins;

Time::sec = t.sec;

return \*this;

}

istream& operator>>(istream& in, Time& t)

{

cout << "Часы: "; in >> t.hours;

cout << "Минуты: "; in >> t.mins;

cout << "Секунды: "; in >> t.sec;

cout << "\n\n";

return in;

}

ostream& operator<<(ostream& out, const Time& t)

{

out << "\n" << t.hours << " ч " << t.mins << " мин " << t.sec << " сек " ;

out << "\n";

return out;

}

void Time::Show()

{

cout << hours << " ч " << mins << " мин "<< sec << " сек " << endl;

}

void Time::operator||(const Time& t) // операция сравнения

{

int a = 0;

if (hours > t.hours) {

a = 1;

}

else if (hours < t.hours) { a = 2; }

else {

if (mins > t.mins) { a = 1; }

else if (mins < t.mins) { a = 2; }

else {

if (sec > t.sec) { a = 1; }

else if (sec < t.sec) { a = 2; }

else { a = 3; }

}

}

if (a == 1) { cout << hours << " ч " << mins << " мин " << sec << " сек > " << t.hours << " ч " << t.mins << " мин " << t.sec << " сек " << endl; }

else if (a == 2) { cout << hours << " ч " << mins << " мин " << sec << " сек < " << t.hours << " ч " << t.mins << " мин " << t.sec << " сек " << endl; }

else { cout << hours << " ч " << mins << " мин " << sec << " сек = " << t.hours << " ч " << t.mins << " мин " << t.sec << " сек " << endl; }

}

**Функция main**

int main() {

setlocale(0, "Rus");

Time t1;

// ввод данных обхекта t1

cin>> t1;

// вывод данных объекта t1

cout << "Объект t1: " << t1 << endl;

// создание объекта t2 с параметрами 12, 30, 10

Time t2(12, 30, 10);

cout << "Объект t2: " << t2 << endl;

// присваивание данных объекта t2 объекту t1

t1 = t2;

cout << "Обновленный объект t1: " << t1;

cout << "\n";

cout << "Сравнение t1 и t2" << endl;

t1 || t2;

cout << "\n";

Time t3(7, 32, 10);

cout << "Объект t3: " << t3 << endl;

cout << "Сравнение t1 и t3" << endl;

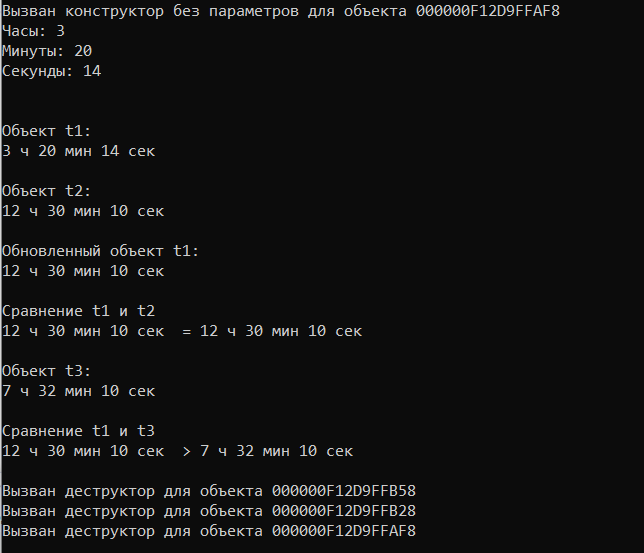
t1 || t3;

cout << "\n";

return 0;

}

**Результат работы программы**



**Ответы на контрольные вопросы**

1. Для чего используется механизм наследования?

Наследование позволяет создавать новые классы, которые повторно используют, расширяют и изменяют поведение, определенное в других классах.

2. Каким образом наследуются компоненты класса, описанные со спецификатором public?

Член класса может использоваться любой функцией, которая является членом данного или производного класса, а также к public-членам возможен доступ извне через имя объекта.

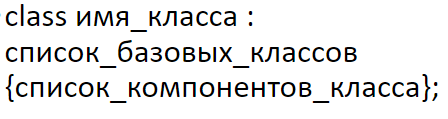
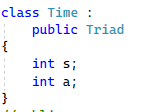
3. Каким образом наследуются компоненты класса, описанные со спецификатором private?

Член класса может использоваться только функциями- членами данного класса и функциями-«друзьями» своего класса. В производном классе он недоступен;

4. Каким образом наследуются компоненты класса, описанные со спецификатором protected?

То же, что и private, но дополнительно член класса с данным атрибутом доступа может использоваться функциями–членами и функциями-«друзьями» классов, производных от данного;

5. Каким образом описывается производный класс?

  
6. Наследуются ли конструкторы?

Не наследуются.

7. Наследуются ли деструкторы?

не наследуются, но они вызываются, когда дочерний класс инициализирует свой объект.

8. В каком порядке конструируются объекты производных классов?

Объекты класса конструируются снизу вверх: сначала базовый, потом компоненты-объекты (если они имеются), а потом сам производный класс. Таким образом объект производного класса содержит в качестве подобъекта объект базового класса.

9. В каком порядке уничтожаются объекты производных классов?

Деструкторы вызываются в обратном порядке от конструкторов.

10. Что представляют собой виртуальные функции и механизм позднего связывания?

К механизму виртуальных функций обращаются в тех случаях, когда в каждом производном классе необходимо, чтобы функции-компоненты работали иначе, чем базовом.

11. Могут ли быть виртуальными конструкторы? Деструкторы?

Конструкторы нет, деструкторы да.

12. Наследуется ли спецификатор virtual?

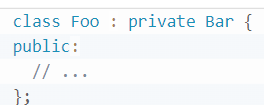
Да

13. Какое отношение устанавливает между классами открытое наследование?

Класс потомок имеет доступ ко всем возможностям класса предка.

14. Какое отношение устанавливает между классами закрытое наследование?

Закрытое наследование означает реализацию посредством - т.е. от базового класса необходимо взять какую-то функциональность, базовый класс и потомок не имеют како-либо концептуальной связи.Закрытое наследование не носит характера отношения подтипов.Закрытое (также как и защищенное) наследование не создает иерархии типов.



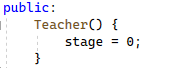
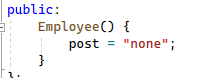
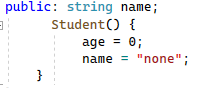
15. В чем заключается принцип подстановки?

Везде, где можно использован объект базового класса, можно использовать вместо него и объект производного.

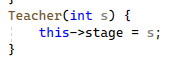
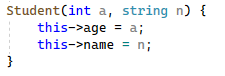
16. Имеется иерархия классов:  
class Student  
{  
int age;  
public:  
};  
string name;  
class Employee : public Student  
{  
protected:  
};  
string post;  
class Teacher public Employee  
{  
protected: int stage;  
};  
Teacher x;  
Какие компонентные данные будет иметь объект х?

Name и Stage, private age не наследуется, post наследуется в private секции.

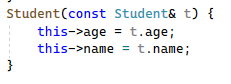
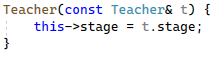
17. Для классов Student, Employee и Teacher написать конструкторы без параметров.



18. Для классов Student, Employee и Teacher написать конструкторы с параметрами.



19. Для классов Student, Employee и Teacher написать конструкторы копирования.

20. Для классов Student, Employee и Teacher определить операцию присваивания.

