Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе № 18.4**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Простое наследование. Принцип подстановки.

Вариант 11

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

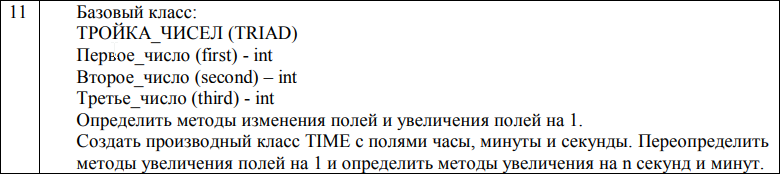
**Пермь, 2021**

**Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Создание иерархии классов с использованием простого наследования.
3. Изучение принципа подстановки.

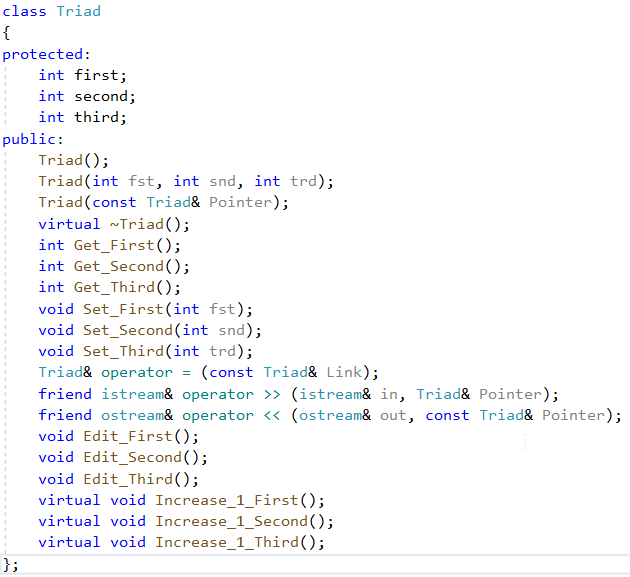
**Постановка задачи**

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 операцию ввода и вывода объектов с помощью потоков.
7. Определить произвольный 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.

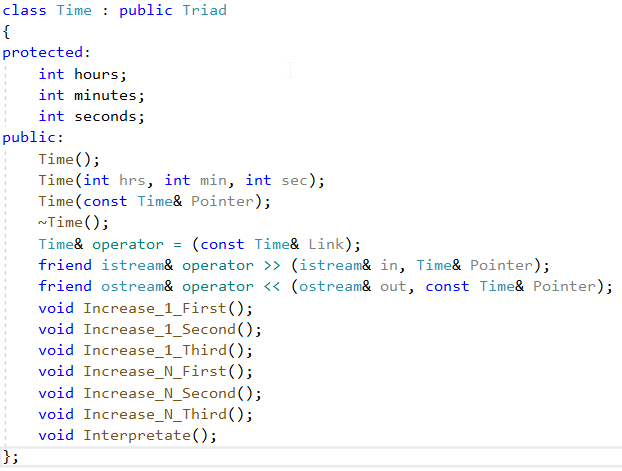


**Анализ задачи**

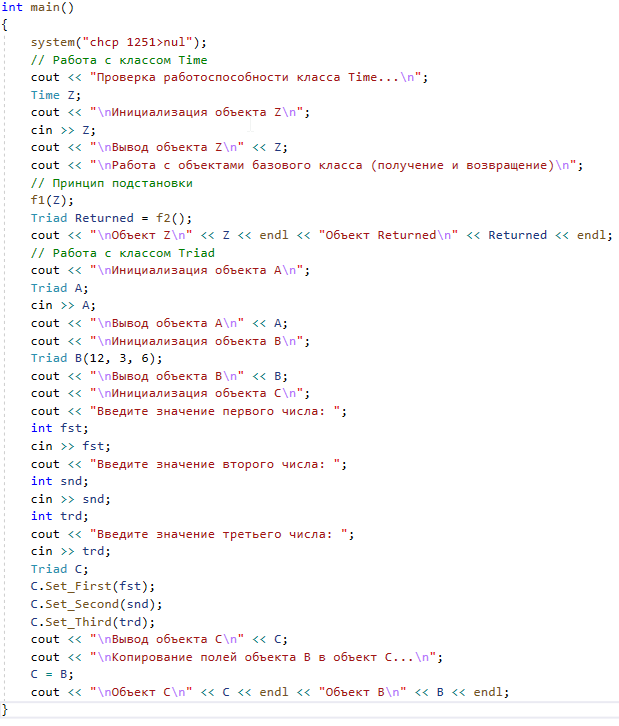
1. Для решения задачи необходимо:
   1. Реализовать определение класса Triad и определение необходимых методов и операторов данного класса.



* 1. Реализовать определение производного от класса Triad класса Time и определение необходимых методов и операторов данного класса.



* 1. Реализовать применение этих функций в главной функции.



1. В ходе работы были использованы типы данных:
   1. Для конструктора без параметров Triad / Time класса Triad / Time не используются аргументы:

Сам конструктор заполняет атрибуты объекта класса “пустотами”

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_YuWCmwH4wC.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_9NAtKdmEcr.png

* 1. Для конструктора с параметрами Triad / Time класса Triad / Time используются следующие аргументы:
  2. Тип int: первое число / количество часов.
  3. Тип int: второе число / количество минут.
  4. Тип int: третье число / количество секунд.

Сам конструктор заполняет атрибуты объекта класса параметрами.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_aPwfPlaAzb.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_txTjaaDy52.png

* 1. Для конструктора копирования Triad / Time класса Triad / Time используются следующие аргументы:
  2. Адресация типа const Triad& / const Time&: объект копирования.

Сам конструктор заполняет атрибуты объекта класса атрибутами объекта копирования.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_BMCTnshI9v.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_PJ1WD1ZH6B.png

* 1. Для деструктора Triad / Time класса Triad / Time не используются аргументы:

Сам деструктор удаляет атрибуты объекта класса.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_hIw587GePm.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_i9K0nq6uwY.png

* 1. Для селектора Get\_First не используются аргументы.

Сам селектор имеет тип int и возвращает значение поля first.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_c1yw1TbJRH.png

* 1. Для селектора Get\_Second не используются аргументы.

Сам селектор имеет тип int и возвращает значение поля second.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_D3JkEcjCYK.png

* 1. Для селектора Get\_Third не используются аргументы

Сам селектор имеет тип int и возвращает значение поля third.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_xUSGZ7NERm.png

* 1. Для модификатора Set\_First используются следующие аргументы:

1. Тип int: первое число.

Сам модификатор имеет тип void, поскольку при работе селектора не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_gdNYAsS4Fp.png

* 1. Для модификатора Set\_Second используются следующие аргументы:

1. Тип int: второе число.

Сам модификатор имеет тип void, поскольку при работе селектора не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_v4dVK1zER9.png

* 1. Для метода Set\_Third используются следующие аргументы:

1. Тип int: третье число.

Сам модификатор имеет тип void, поскольку при работе селектора не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_4daTkteBmD.png

* 1. Для метода Increase\_1\_First не используются аргументы.

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_7cBXMj4bJq.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_OsL8ZDse2d.png

* 1. Для метода Increase\_1\_Second не используются аргументы.

Сам метода имеет тип void, поскольку при работе метода не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_Ywu09GqJKV.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_Sur4h5EDNl.png

* 1. Для метода Increase\_1\_Third не используются аргументы.

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_qAswadxE5P.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_5kJwKNAK1l.png

* 1. Для метода Increase\_N\_First не используются аргументы.

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_qhBfGO7wyP.png

* 1. Для метода Increase\_N\_Second не используются аргументы

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_pc4mwkd1oW.png

* 1. Для метода Increase\_N\_Third не используются аргументы.

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_bb46dtgkbc.png

* 1. Для метода Edit\_First не используются аргументы.

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_hQtOfyhuR8.png

* 1. Для метода Edit\_Second не используются аргументы.

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_I4rYghcaXo.png

* 1. Для метода Edit\_Third не используются аргументы.

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_7qFrGf6Nxk.png

* 1. Для метода Interpretate не используются аргументы.

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_mhGrrF74vX.png

* 1. Для метода перегрузки оператора присваивания используются следующие аргументы:
  2. Тип const Triad& / Time&: ссылка на объект.

Сам метод имеет тип Triad& / Time& и возвращает указатель на объект this.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_rC0RvdB3q9.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_GAEEplHtpY.png

* 1. Для дружественной функции перегрузки оператора ввода используются следующие аргументы:
  2. Тип istream&: поток ввода.
  3. Тип const Triad& / const Time&: ссылка на объект.

Сама функция имеет тип istream& и возвращает вводимые данные.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_bcYN5DkIaM.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_DjUTAcTkFp.png

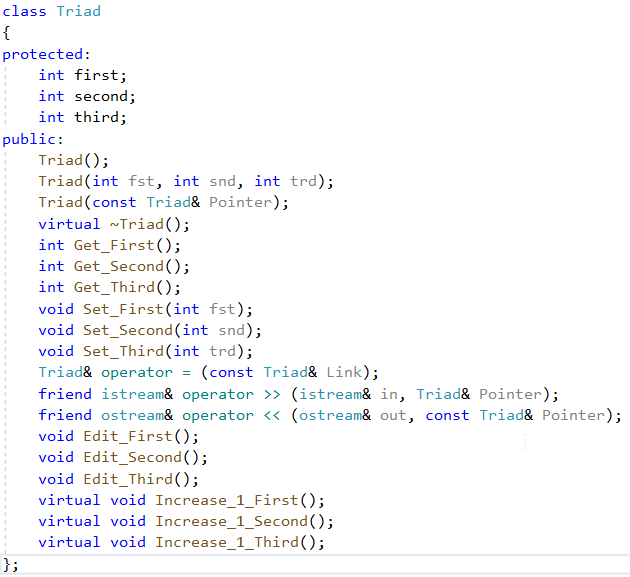
* 1. Для дружественной функции перегрузки оператора вывода используются следующие аргументы:
  2. Тип ostream&: поток вывода.
  3. Тип const Triad& / const Time&: ссылка на объект.

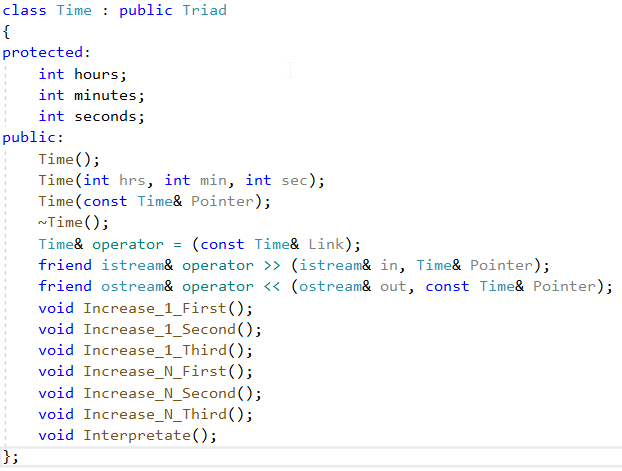
Сама функция имеет тип ostream& и возвращает выводимые данные.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_yfnh7sIjGt.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_7fnNwR9I7W.png

1. Для решения задачи данные были представлены в следующем виде:
   1. Для работы с данными используются атрибуты класса.





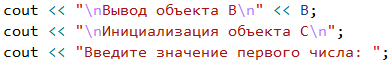
1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Ввод данных реализован с помощью оператора cin, используемых при реализации в главной функции и функций перегрузки оператора.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_8HJRYBdhJf.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_dHykjlPZuD.png

* 1. Вывод данных реализован с помощью оператора cout, используемого при реализации в главной функции и функций перегрузки операторов ввода-вывода.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_cjURuT5J8l.png



1. Поставленные задачи решены следующими действиями:
   1. Определение класса Triad / Time было реализовано в заголовочном файле Triad.h / Time.h, определение конструкторов, деструктора, селекторов и модификаторов класса – в файле Triad.cpp / Time.cpp. Основной блок программы описан в главном файле Лабораторная работа № 18.4.cpp.
   2. При работе с объектами классов Triad и Time обращение к конструктору без параметра аналогично выделению памяти под переменную. Обращение к конструктору с параметрами аналогично обращению к методу Init. Обращение к конструктору копирования аналогично оператору присваивания. Обращение к селекторам и модификаторам аналогично обращению к методам или к полям структуры.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_AZ4Zsn9s5Z.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_NLOGejua2O.png

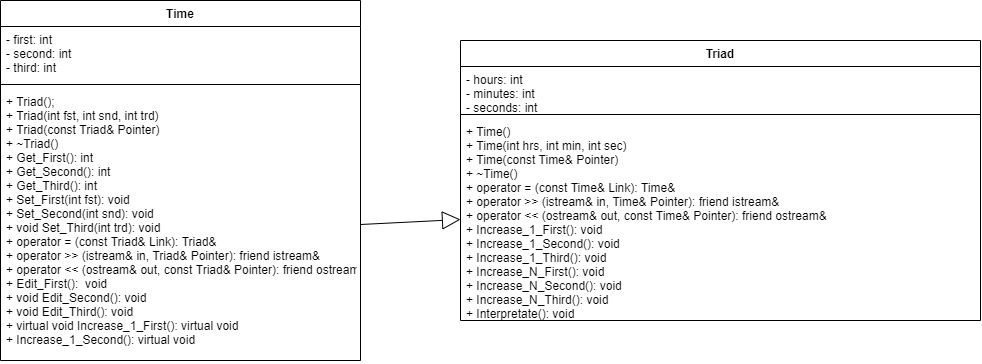
C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_irWHM3Aehc.png

* 1. При работе с перегруженными операторами работа без объектов созданного класса происходит по умолчанию. Если при вызове перегруженных операторов используются объекты класса, то операторы работают согласно их определению в этом классе.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_6xyqtpng5f.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_a8n0a3sgh3.png

**Диаграмма классов**

****

**Реализация задачи на языке С++**

**Заголовочный файл Triad.h**

#pragma once

#include <iostream>

using namespace std;

class Triad

{

protected:

int first;

int second;

int third;

public:

Triad();

Triad(int fst, int snd, int trd);

Triad(const Triad& Pointer);

virtual ~Triad();

int Get\_First();

int Get\_Second();

int Get\_Third();

void Set\_First(int fst);

void Set\_Second(int snd);

void Set\_Third(int trd);

Triad& operator = (const Triad& Link);

friend istream& operator >> (istream& in, Triad& Pointer);

friend ostream& operator << (ostream& out, const Triad& Pointer);

void Edit\_First();

void Edit\_Second();

void Edit\_Third();

virtual void Increase\_1\_First();

virtual void Increase\_1\_Second();

virtual void Increase\_1\_Third();

};

**Файл с описанием методов класса Triad.cpp**

#include "Triad.h"

#include <iostream>

using namespace std;

Triad::Triad()

{

first = 0;

second = 0;

third = 0;

}

Triad::Triad(int fst, int snd, int trd)

{

first = fst;

second = snd;

third = trd;

}

Triad::Triad(const Triad& Pointer)

{

first = Pointer.first;

second = Pointer.second;

third = Pointer.third;

}

Triad::~Triad()

{

}

int Triad::Get\_First()

{

return first;

}

int Triad::Get\_Second()

{

return second;

}

int Triad::Get\_Third()

{

return third;

}

void Triad::Set\_First(int fst)

{

first = fst;

}

void Triad::Set\_Second(int snd)

{

second = snd;

}

void Triad::Set\_Third(int trd)

{

third = trd;

}

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

{

if (&Link != this)

{

first = Link.first;

second = Link.second;

third = Link.third;

}

return \*this;

}

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

{

cout << "Введите значение первого числа: ";

in >> Pointer.first;

cout << "Введите значение второго числа: ";

in >> Pointer.second;

cout << "Введите значение третьего числа: ";

in >> Pointer.third;

return in;

}

ostream& operator << (ostream& out, const Triad& Pointer)

{

cout << "Триада чисел:\n";

out << "Певрое число: " << Pointer.first << endl;

out << "Второе число: " << Pointer.second << endl;

out << "Третье число: " << Pointer.third << endl;

return out;

}

void Triad::Edit\_First()

{

cout << "Введите новое значение первого числа: ";

int fst;

cin >> fst;

first = fst;

}

void Triad::Edit\_Second()

{

cout << "Введите новое значение второго числа: ";

int snd;

cin >> snd;

second = snd;

}

void Triad::Edit\_Third()

{

cout << "Введите новое значение третьего числа: ";

int trd;

cin >> trd;

third = trd;

}

void Triad::Increase\_1\_First()

{

first++;

}

void Triad::Increase\_1\_Second()

{

second++;

}

void Triad::Increase\_1\_Third(){

third++;

}

**Заголовочный файл Time.h**

#pragma once

#include <iostream>

#include "Triad.h"

using namespace std;

class Time : public Triad

{

protected:

int hours;

int minutes;

int seconds;

public:

Time();

Time(int hrs, int min, int sec);

Time(const Time& Pointer);

~Time();

Time& operator = (const Time& Link);

friend istream& operator >> (istream& in, Time& Pointer);

friend ostream& operator << (ostream& out, const Time& Pointer);

void Increase\_1\_First();

void Increase\_1\_Second();

void Increase\_1\_Third();

void Increase\_N\_First();

void Increase\_N\_Second();

void Increase\_N\_Third();

void Interpretate();

};

**Файл с описанием методов класса Time.cpp**

#include "Time.h"

#include <iostream>

using namespace std;

Time::Time()

{

hours = 0;

minutes = 0;

seconds = 0;

}

Time::Time(int hrs, int min, int sec)

{

hours = hrs;

minutes = min;

seconds = sec;

}

Time::Time(const Time& Pointer)

{

hours = Pointer.hours;

minutes = Pointer.minutes;

seconds = Pointer.seconds;

}

Time::~Time()

{

}

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

{

if (&Link != this)

{

hours = Link.hours;

minutes = Link.minutes;

seconds = Link.seconds;

}

return \*this;

}

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

{

cout << "Введите значение часов: ";

in >> Pointer.hours;

while (Pointer.hours < 0 || Pointer.hours > 23)

{

cout << "Введено неправильно число часов. Введите новое значение часов: ";

cin >> Pointer.hours;

}

cout << "Введите значение минут: ";

in >> Pointer.minutes;

while (Pointer.minutes < 0 || Pointer.minutes > 59)

{

cout << "Введено неправильно число минут. Введите новое значение минут: ";

cin >> Pointer.minutes;

}

cout << "Введите значение секунд: ";

in >> Pointer.seconds;

while (Pointer.seconds < 0 || Pointer.seconds > 59)

{

cout << "Введено неправильно число секунд. Введите новое значение секунд: ";

cin >> Pointer.seconds;

}

return in;

}

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

{

cout << "Время\n";

out << "Часов: " << Pointer.hours << endl;

out << "Минут: " << Pointer.minutes << endl;

out << "Секунд: " << Pointer.seconds << endl;

return out;

}

void Time::Increase\_1\_First()

{

hours = (hours + 1) % 24;

}

void Time::Increase\_1\_Second()

{

minutes++;

hours = (hours + minutes / 60) % 24;

minutes = minutes % 60;

}

void Time::Increase\_1\_Third()

{

seconds++;

minutes = minutes + seconds / 60;

hours = (hours + minutes / 60) % 24;

minutes = minutes % 60;

seconds = seconds % 60;

}

void Time::Increase\_N\_First()

{

int N;

cout << "Введите число, на которое хотите увеличить количество часов: ";

cin >> N;

hours = (hours + N) % 24;

}

void Time::Increase\_N\_Second()

{

int N;

cout << "Введите число, на которое хотите увеличить количество минут: ";

cin >> N;

minutes += N;

hours = (hours + minutes / 60) % 24;

minutes = minutes % 60;

}

void Time::Increase\_N\_Third()

{

int N;

cout << "Введите число, на которое хотите увеличить количество секунд: ";

cin >> N;

seconds += N;

minutes = minutes + seconds / 60;

hours = (hours + minutes / 60) % 24;

minutes = minutes % 60;

seconds = seconds % 60;

}

void Time::Interpretate()

{

hours = first % 24;

minutes = second % 60;

seconds = third % 60;

}

**Файл с главной программой Лабораторная работа № 18.4.cpp**

#include <iostream>

#include "Triad.h"

#include "Time.h"

using namespace std;

void f1(Triad& Pointer)

{

Pointer.Set\_First(145);

Pointer.Set\_Second(329);

Pointer.Set\_Third(87);

cout << Pointer;

}

Triad f2()

{

Time Y(23, 15, 34);

return Y;

}

int main()

{

system("chcp 1251>nul");

// Работа с классом Time

cout << "Проверка работоспособности класса Time...\n";

Time Z;

cout << "\nИнициализация объекта Z\n";

cin >> Z;

cout << "\nВывод объекта Z\n" << Z;

cout << "\nРабота с объектами базового класса (получение и возвращение)\n";

// Принцип подстановки

f1(Z);

Triad Returned = f2();

Time

cout << "\nОбъект Z\n" << Z << endl << "Объект Returned\n" << Returned << endl;

// Работа с классом Triad

cout << "\nИнициализация объекта A\n";

Triad A;

cin >> A;

cout << "\nВывод объекта A\n" << A;

cout << "\nИнициализация объекта B\n";

Triad B(12, 3, 6);

cout << "\nВывод объекта B\n" << B;

cout << "\nИнициализация объекта C\n";

cout << "Введите значение первого числа: ";

int fst;

cin >> fst;

cout << "Введите значение второго числа: ";

int snd;

cin >> snd;

int trd;

cout << "Введите значение третьего числа: ";

cin >> trd;

Triad C;

C.Set\_First(fst);

C.Set\_Second(snd);

C.Set\_Third(trd);

cout << "\nВывод объекта C\n" << C;

cout << "\nКопирование полей объекта B в объект C...\n";

C = B;

cout << "\nОбъект C\n" << C << endl << "Объект B\n" << B << endl;

}

**Ответы на вопросы**

1. Для чего используется механизм наследования?

Наследование − это механизм получения нового класса на основе уже существующего

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

Член класса может использоваться любой функцией, которая является членом данного или производного класса.

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

В производном классе он недоступен.

1. 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.

1. Каким образом описывается производный класс?

class имя\_класса : список\_базовых\_классов

{список\_компонентов\_класса};

1. Наследуются ли конструкторы?

Нет.

1. Наследуются ли деструкторы?

Нет.

1. В каком порядке конструируются объекты производных классов?

Базовый класс -> компоненты-объекты -> производный класс

1. В каком порядке уничтожаются объекты производных классов?

Производный класс -> компоненты-объекты -> базовый класс

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

Виртуальные функции предоставляют механизм позднего (отложенного) или динамического связывания.

1. Могут ли быть виртуальными конструкторы? Деструкторы?

Конструкторы – нет, деструкторы – да.

1. Наследуется ли спецификатор virtual?

Да.

1. Какое отношение устанавливает между классами открытое наследование?

Открытое наследование устанавливает между классами отношение «является»: класс-наследник является частью класса-родителя.

1. Какое отношение устанавливает между классами закрытое наследование?

Закрытое наследование – это наследование реализации, в этом случае принцип подстановки не соблюдается.

1. В чем заключается принцип подстановки?

Везде, где может быть использован объект базового класса, вместо него разрешается использовать объект производного класса.

1. 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, post, stage.

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

Student()

{  
age = 0;

Name = “”;

}

Employee() : Student()

{  
post = “”;

}

Teacher() : Employee()

{  
stage = 0;

}

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

Student(int Age, string Name)

{  
age = Age;

name = Name;

}

Employee(string Post) : Student(Age, Name)

{  
post = Post;

}

Teacher(int Stage) : Employee(Post)

{  
stage = Stage;

}

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

Student(const Student& elem)

{  
age = elem.age;

name = elem.name;

}

Employee(const Employee& elem) : Student(elem)

{  
post = elem.post;

}

Teacher(const Teacher& elem) : Employee(elem)

{

stage = elem.stage;

}

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

Student& operator =(const Student& elem)

{

if(&elem==this)return \*this;  
age = elem.age;

name = elem.name;

return \*this;

}

Employee& operator =(const Employee& elem) : Student(elem)

{

if(&elem==this)return \*this;  
post = elem.post;

return \*this;

}

Teacher& operator =(const Teacher& elem) : Employee(elem)

{

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

stage = elem.stage;

return \*this;

}