**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 № 19.5**

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

**Ответы на вопросы:**

1. Что послужило толчком к созданию парадигмы ООП?

Толчком к созданию парадигмы послужил кризис программного обеспечения (высокая сложность программ в промышленности, координирование команды разработчиков, необходимость в сопровождении про­ граммы), трудность в достижении слабой связанности в структурной декомпозиции, проблема общего глобального пространства имён.

1. Экскурс в историю: в каком языке были разработаны основные прин­ ципы ООП? Дайте краткий обзор этого языка.

Первым объектно­ориентированным языком был язык программирования Симула, который был создан в компьютерном центре в Норвегии. Язык предназначался для моделирования ситуаций реального мира. Особенностью Simula было то, что программа, написанная на языке, была организована по объектам программирования. Объекты имели инструкции, называемые методами, и данные, которые назывались переменными; методы и данные определяли поведение объекта. В процессе моделирования объект вел себя согласно своему стандартному поведению и, в случае необходимости, изменял данные для отражения влияния назначенного ему действия.

1. Подробно опишите показатели качества декомпозиции программы Показателями качества декомпозиции программы являются:

Сцепление внутри компонента это показатель, характеризующий степень взаимосвязей отдельных его частей (если внутри компонента решаются 2 подзадачи, которые легко можно разделить, то компонент обладает слабым, плохим сцеплением) и связанность между компонента­ ми (это показатель описывающий интерфейс между компонентом клиентом и компонентом сервером). Общее число входов и выходов сервера ­ есть мера связанности. Чем меньше связанность между двумя компонентами, тем проще поменять и отслеживать в будущем их взаимодействия, а так как в больших проектах эти компоненты часто разрабатывается разными людьми, то очень важно уменьшать связанность между компонентами.

1. Назовите несколько отличий языков C и С++.
   * В C++ появились классы и объекты.
   * В С символьная константа автоматически повышается до целого числа. В С++ это не так.
   * В С можно вызвать функцию main() из программы, хотя это и нетипично. В С++ это недопустимо.
   * Разные стандартные расширения файлов, .c для С, и .cpp С++.
2. Подробно опишите идентификаторы (где начало и конец области дей­ ствия, какова видимость, каково время жизни), описанные в блоке, фай­ ле, функции, прототипе функции, классе и поименованной области.

В блоке:

* + Начало области действия идентификатора, когда происходит объ­ явление в блоке
  + Конец области действия идентификатора, когда заканчивается блок
  + Видимость только в блоке
  + Время жизни, пока работает программа через блок. В файле:
  + Начало области действия идентификатора при объявлении в функции
  + Конец области действия идентификатора, когда заканчивается про­ грамма
  + Видимость во всем, но не в тех функциях, куда не передаётся
  + Время жизни , пока работает программа В функции:
  + Начало области действия идентификатора также при объявлении
  + Конец области действия идентификатора, когда заканчивается функ­ ция
  + Виден после того, как объявили в программе
  + Время жизни, пока исполняется, то есть, когда вызывают В прототипе функции:
  + Начало области действия идентификатора также при объявлении
  + Конец области действия идентификатора, когда заканчивается функ­ ция
  + Виден после того, как объявили в программе
  + Время жизни, пока исполняется, то есть, когда вызывают
  + (используется для обозначения существования для компилятора до тех пор, пока еще не определили функцию)

В классе:

* + Начало области действия идентификатора, когда создается объект, в котором используется
  + Конец области действия идентификатора, при удалении объекта
  + Виден везде, при условии, что создан в основном теле программы
  + Время жизни, пока объект продолжает существование в програм­ ме

В поименованной области:

* + Начало области действия идентификатора, когда происходит объ­ явление
  + Конец области действия идентификатора , когда заканчивается про­ грамма
  + Виден везде после объявления
  + Время жизни до конца программы

1. Чем отличаются заголовочные файлы stdio.h и cstdio?

stdio.h используется в С, а cstdio в С++, в современных библиотеках имеются более безопасные аналоги «классических» функций.

1. В чём структурный подход к программированию уступает объектно­ ориентированному подходу?
   * повышение степени абстракции программы
   * ослабление связанности между компонентами программы
   * повышение надёжности программ
   * облегчение модификации программы
   * свойства не описываются повторно

**Код с идентификаторами:**

1. main.cpp

#include <iostream>  
using namespace std;  
  
class one {  
public:  
 int sample = 5; // 1  
 int func(int sample) { // 2  
 return ++sample;  
 }  
};  
  
namespace two {  
 int sample() { // 3  
 return 5;  
 }  
  
 int sample(int sample) { // 4 5  
 return ++sample;  
 }  
}  
  
class three {  
 int sample = 6; // 6  
 int func(int sample) { // 7  
 return ++sample;  
 }  
};  
  
int main() {  
 one sample; // 8  
 cout << sample.func(sample.sample) << endl;  
 for (int sample = 0; sample < 5; sample++) // 9  
 cout << "Hello" << endl;  
 return 0;  
}