

Содержание

1	Лекция 1	2
2	Лекция 2	4
3	Лекция 4	7
4	Лекция 5	7
5	Лекция 6	8
6	Лекция 7	11

1 Лекция 1

Требования:

Списывание строго карается.

1-4 задания на C++ остальные лабораторные работы на шарпе.

На все задания есть строки (если все они выполнены, то возможен автомат).

На форуме есть методичка (где-то).

У преподавателя из методичек списывать можно.

Литература:

matanit (сайт)

Преди Буч "введение в объектно ориентированное программирование"

Джон фон Нейман создал основную архитектуру процессора.

Чарльз Бэббидж — первый компьютер.

Эниак 1944 первый компьютер с разделением памяти.

1940-1950 все писали на машинном коде. (языков программирования ещё нет)

± 50-е годы появляются ассемблеры и Алгол. (1-е поколение языков)

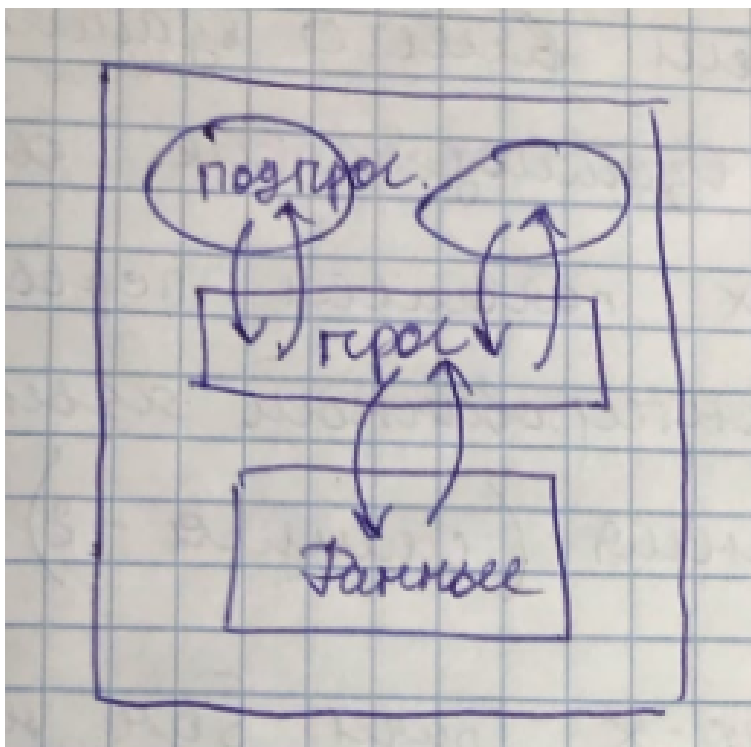
?Мнемоническая команда?

В ассемблере появилась формальная адресация, в алголе появились первые высокоуровневые команды.

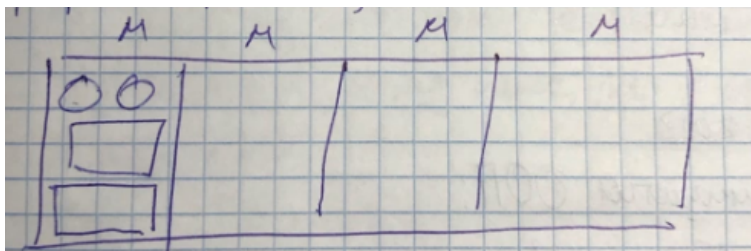
Любой 16-тиричный код ассемблер рассматривает как команды.

В алголе впервые появилось жёсткое деление на программу и данные.

Во втором поколении языков появились функции и подпрограммы + появились локальные и глобальные переменные. (алгол, фор-тран, ада)



После этого появилось деление на модули (для возможности совместного программирования).



Связь между модулями осуществляется строго интерфейсами или протоколами, прямого доступа нет. (3-е поколение языков)

Дальше ввели вместо взаимодействия алгоритмов, взаимодействие объектов.

В середине 60-х появился первый объектно ориентированный язык программирования (симула-2)

С середины 80-х Гради БУч начинает развивать ООП.

?Принцип абстракции в программировании?

2 Лекция 2

Принципы ООП:

- 1) Абстрагирование.
- 2) Инкапсуляция (регулировка доступа к характеристикам и поведению объекта)

1. Проблема метода заключается в возможности злоумышленного изменения характеристик.

```
struct cat
{
    string poroda;
    color cEye;
    string name;
    int weight;
};
```

2. Преимущество по сравнению с первым вариантом в защищённости характеристик.

```
class cat
{
private:    // личная часть класса
    string poroda;
    int cEye;
    string name;
    int weight;

    void changeWeight() //защищённый доступ к изменению характеристики
    {
        .....
    }

public: // общедоступная часть класса
    cat(string p, int ceYe, string n, int w) // конструктор класса
    {
        poroda = p;
        cEye = ceYe;
        name = n;
        weight = w;
    }
    string getP() //getter
    {
        return poroda;
    }
    int getEye()
```

```

{
    return cEye;
}
string getName()
{
    return name;
}
void setName(string n) //сеттер
{
    name = n;
}
int getWeight()
{
    return weight;
}
void Eat()
{
    changeWeight();
}
};

```

Если действие может привести к разрушению объекта, то, как правило, доступ к нему ограничивают (размещают в private) + характеристики (все) обычно не хранятся в общем доступе.

Пользователь = вредитель, поэтому максимально защищайтесь и ограничивайте доступ.

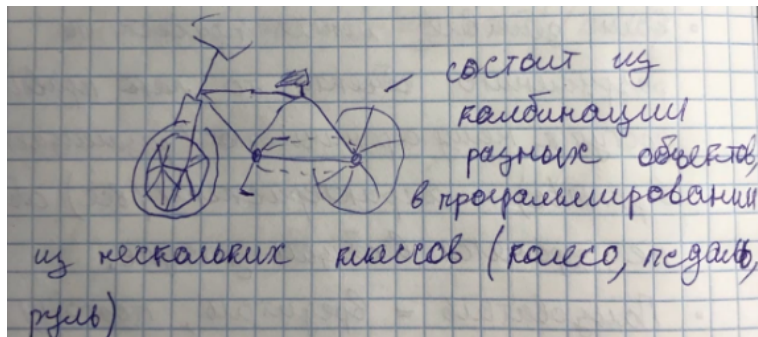
?protected?

?published?

Иерархия бывает двух видов:

- 1) part of (это часть того).
- 2) is as (это разновидность того)

part of экономит на написании кода, так как выделяет общий набор характеристик и действий, которые объекты могут совершать.



4.

```
(is as)
class Ep
{
private:
    string marka;
    int power;
    int voltage;
public:
    Ep(){}
    void on(){}
    void off(){}
    void print(){}
};

/* удобно возможностью разом совершать какие-то действия +
экономию кода + возможность создания классов наследования
(в данном случае классов электроприборов, например, класс телевизора)
*/
class TV : public Ep
{
private:
    int diag;
    string matrix;
public:
    TV() : Ep(){} //конструктор класса
    void print(){}
};
```

Если сделать следующим образом

```
E.print();
t.print();
f.print();
E=t;
E.print();
```

то выведется метод print для E, а не для t

Если на Ер указан **virtual** (виртуальный класс), то при

```
Er* ee;  
ee = &t;  
ee->print();
```

то используется метод print для t, если оставить

```
E.print();  
t.print();  
f.print();  
E=t;  
E.print();
```

то результат останется прежним.

3 Лекция 4

В питоне есть множественное присвоение, но важен порядок наследования, так как пересекающиеся методы у классов родителей будут взяты у первого в очереди наследования.

Батраева не уважает python.

Батраева любит, когда её слушают.

В с++ попарная реализация (`==` и `!=`), (`>` и `<`) и (`>=` и `<=`), то есть достаточно определить 1 элемент из пары.

В языках появившихся после с++, как правило, объект является указателем и динамическим объектом.

Безопасное преобразование в сишарпе реализуется через тернарный оператор.

4 Лекция 5

Абстрактные классы нужны для интерфейсов.

Интерфейс — метод взаимодействия с другими объектами.

Абстрактный класс нужен для создания общего порядка.

В subbuilder и дельфи есть общий родительский класс ToObject.

При большой глубине наследования сильно затрудняется отладка, нужно быть осторожным.

В с++ есть возможность функцию передать как параметр.

В с++ есть возможность создать статическую переменную, её значение видят все объекты класса.

Так же она выступает в качестве контроля объектов класса в языках с динамическими классами (пример: фокус ввода в ОС).

Важно: раньше сапёра можно было открыть только в 1 экземпляре!!!

5 Лекция 6

Жизненный цикл:

1. Организационные проекты
 - (a) Управление проектом
 - (b) Создание инфраструктуры
 - (c) Оценка и улучшение жизненного цикла
 - (d) Обучение
2. Основные процессы
 - (a) Приобретение
 - (b) Поставка
 - (c) Разработка
 - (d) Эксплуатация
 - (e) Сопровождение
3. Вспомогательные процессы
 - (a) Документирование
 - (b) Управление конфигурацией
 - (c) Обеспечение качества
 - (d) Верификация
 - (e) Аттестация
 - (f) Совместная оценка
 - (g) Аудит
 - (h) Разрешение проблем

В компаниях, плохо работающих с ресурсами, лучше работать только в целях набора опыта.

Сотрудника могут принять либо в штат, либо на проект (пока идёт проект). Второй вариант используют компании, как правило, плохо управляющие проектом.

В жизненном цикле важно следить за совместимостью ПО с железом.

Хорошее внедрение лучше хорошего разработчика. Внедрение очень важно!

70 лет = очень старый.

Если вы работаете с облаками важно иметь разрешение на эксплуатацию в нём (в договоре).

Верификация, т.е. в принципе программа работает верно.

Обеспечение качества = проверка по ТЗ.

На аудит могут приглашаться сторонние оценщики.

ПО в компаниях с жёсткими требованиями к безопасности не должно иметь незадокументированных дополнительных возможностей.

Заказчиком принимаются следующие действия:

Осознание своих потребностей в программной системе и принятие решения относительно покупки, разработки под заказ или усовершенствования существующей системы.

Подготовка заявочных предложений, содержащих требования к системе, условия её функционирования и технические ограничения, а также другие условия контракта.

Процесс разработки включает в себя:

Оформление проектной и эксплуатационной документации.

Подготовку материалов, необходимых для проверки работоспособности программного продукта и его соответствия стандартам качества.

Разработку материалов необходимых для подготовки и обучения персонала.

Выбор модели жизненного цикла.

Анализ требований к системе.

Проектирование архитектуры системы.

Анализ программных требований.

Детальное конструирование ПО.

Конструирование и тестирование ПО.

Интеграция ПО.

Квалификационное тестирование ПО.

Интеграция системы.

Квалификационное тестирование системы.

Установка ПО.

Приёмка ПО.

Анализ требований к системе:

На данном этапе рассматривается область применения системы. Список требований к разрабатываемой системе должен включать.

Совокупность условий, при которых предполагается эксплуатировать будущую систему.

Описание выполняемых системой функций.

Ограничения в процессе разработки.

Анализ требований к ПО:

Предполагается определение следующих характеристик для каждого компонента ПО:

Функциональных возможностей, включая характеристики производительности и среды функционирования компонента.

Внешних интерфейсов.

Спецификация надёжности и безопасности.

Эргономических требований.

Требований к используемым данным.

Требований к установке и приёме.

Требований к пользовательской документации.

Требований к эксплуатации и сопровождению.

Функциональные возможности говорит заказчик, характеристики производительности тоже, желательно, сообщает заказчик.

Если что-то стирается из БД, то лучше потребовать нормативы по хранению данных организации.

Наработка на отказ — время работы без перерыва, идеал 99,9 процентов часов работы за год.

Эргономические требования = вопрос об интерфейсах.

Презентация строго чёрными буквами на белом фоне, если у Батраевой попадёте, то "затопчит и уничтожит".

Интерфейсы должны быть стандартизированными.

Документация по ПО должна быть исчерпывающим для пользователя.

У SQL сервера microsoft лучшая, по мнению Батраевой, документация.

В рамках проектирования архитектуры для каждого компонента ПО выполняются следующие задачи:

Определение на высоком уровне абстракции структуры программного обеспечения и состава его компонентов.

Разработка и документирование программных интерфейсов ПО и баз данных.

Разработка предварительной версии пользовательской документации.

Разработка и документирование предварительных требований к тестам и планам интеграции ПО.

Детальное конструирование ПО включает в себя:

Описание компонентов ПО и интерфейсов между ними в объёме достаточном для их последующего самостоятельного кодирования и тестирования.

Разработку и документирование детального проекта базы данных.

Обновление пользовательской документации.

Разработку и документирование требований к тестам и плана тестирования компонентов.

Обновление плана интеграции ПО.

Квалификационное тестирование ПО проводится разработчиком в присутствии заказчика для демонстрации того, что ПО удовлетворяет своим спецификациям и готово к использованию в условиях эксплуатации.

При этом также проверяется полнота технической и пользовательской документации и её адекватность компонентам ПО.

Установка ПО осуществляется разработчиком в соответствии с планом в той среде и на том оборудовании, которые предусмотрены договором. В процессе установки проверяется работоспособность ПО и баз данных.

Приёмка ПО предусматривает оценку результата квалификационного тестирования системы и документирование результатов оценки, которое производится заказчиком с помощью разработчика. Разработчик выполняет окончательную передачу ПО заказчику в соответствии с договором, обеспечивая при этом необходимое обучение и поддержку.

Функции службы сопровождения:

Анализ проблем и запросов на модификацию ПО.

Модификация программного продукта.

Его проверка и приёмка.

Перенос ПО в другую среду.

Снятие ПО с эксплуатации.

Снятие ПО с эксплуатации = архивизация данных в читабельный формат + стирание программы.

6 Лекция 7

Документирование — формализованное описание информации, созданной в течении всего жизненного цикла ПО.

Конфигурация — это совокупность функциональных и физических характеристик, установленных в технической документации и реализованных в программе.

Качество — совокупность свойств, характеризующих способность ПО удовлетворять требованиям.

Верификация — совокупность действий по сравнению полученного результата жизненного цикла с требуемым результатом.

Аттестация — определение полноты соответствия ПО функциональному назначению.

Модели жизненного цикла:

1. Каскадная (этапы строго последовательные).
2. Инкрементная (выполняются последовательно маленькие части этапов)