Дисклеймер.

Автор не несет ответственности за любой ущерб, причиненный Вам при использовании данного документа. Автор напоминает, что данный документ может содержать ошибки и опечатки, недостоверную и/или непроверенную информацию. Если Вы желаете помочь в развитии проекта или сообщить об ошибке/опечатке/неточности:

GitHub проекта

Автор в ВК

Аннотация

«Сюда не ходи, снег в башка попадёт - совсем мертвым будешь!» ©Терехов

1 ООП. Проектирование, инкапсуляция и скрытие информации; разделение интерфейса и реализации; классы, наследники, наследование; полиморфизм; иерархии классов.

Класс — тип, определенный пользователем. Компоненты, использованные при определении класса — его члены.

Подпрограмма — стек, $A \Rightarrow B \Rightarrow C$ и обратно. А сопрограмма — куча с произвольным взаимодействием.

Объект —

Симула67 — первый ОО язык. Затем SmallTalk — первый нормальный ОО язык.

Объектное программирование отличается от обычного тем, что имеются некие объекты, которые «носят с собой» все свои данные.

Инкапсуляция — сокрытие данных (хороший пример — стек. Сам он реализован черт знает как, а «торчат наружу» только PUSH и POP).

```
class
{

    int a;
    int b;
    ---

}
```

Пример класса:

```
class X
{

    public:
        интерфейс;
    private:
        реализация;

}
```

Конструктор класса:

```
struct Date
{

    int y, m, d;
    Date(int y, int m, int d);
```

```
}
```

Большие функции лучше определять вне класса:

```
Date :: Date (int y, int m, int d)
{
};
```

Здесь мы указываем, что функция Date является членом класса Date.
Наследование:

```
struct B
{

    int mb;
    void fb()
        {
        ---
        };

}
class D : B
{

    int mb;
    void fb()

}
class DD:public B1, private B2
{
---
}
```

Public — доступ открыт всем, кто видит определение данного класса.
Private — доступ открыт самому классу и друзьям данного класса.
Protected — доступ открыт классам, производным от данного.

2 Основные вычислительные алгоритмы: алгоритмы поиска и сортировки.

3 Основы программирования, основанного на событиях

Система реального времени — система, которая должна обмениваться данными за какой-либо промежуток времени.

Следует различать сбой и отказ. В среднем на 10000 сбоев случается 1 отказ.

Гомогенная схема — элементы, равные по ответственности и надежности. Они дешевы, но если один из элементов врет, то в схеме из двух элементов нельзя определенно

сказать, который. А вот если схема из 3 элементов, (2oo3), то «есть кого обматерить». Одновременно два выйти из строя могут с ничтожно малой вероятностью.

ОВК — отказоустойчивый военный комплекс.

Нужно сказать, что такое Mission critical system, что они управляются особо надежными системами. Дублированные комплексы не годятся по причине низкой отказоустойчивости.

SDL — графический язык (1984).

Событие — приход сигнала.

4 Введение в компьютерную графику.

Рассказывал Брыксин.

5 Обзор языков программирования: история языков программирования; краткий обзор парадигм программирования.

Сначала люди писали в машинных кодах. Но быстро поняли, что это говно, легко ошибиться, и перешли к языкам ассемблера. Минусы ассемблера в том, что мы можем сложить две ячейки памяти вне зависимости от того, что там находится. Таким образом, ассемблер является незащищенным языком.

В 1956 году фирмой IBM представлен язык FORTRAN. Этот язык уже похож на современные языки программирования: в нем были операции, типы данных и так далее.

Самая дорогая ошибка мира:

```
do 3 i=1.4
```

Вместо точки должна была быть запятая. Написал ошибочку. А это была программа, управляющая полетом ракеты на Венеру. Из-за ошибки 7 миллиардов долларов улетели в трубу. Виновным признали язык Фортран и было решено придумать новый, более хороший язык. В устроенном конкурсе победил язык Ада, названный в честь Ады Лавлейс.

В 1958 году в Европе «в ответ Америке» был придуман язык АЛГОЛ-60. Его начали улучшать. Улучшения происходили в 60, 64, 68 и 74 году. Лучшими были признаны АЛГОЛ-60 и АЛГОЛ-68.

Были различные логические языки, но выжил только один, ПРОЛОГ. Синтаксис:

```
Логическое утверждение:
A здесь выводы.
```

В общем, все логические языки сдохли, а ПРОЛОГ выжил, благодаря возможности управления порядком применения логических правил. Автор первой реализации ПРОЛОГа — поляк, а хорошая версия ПРОЛОГа была сделана в Греции.

Короче, как-то один мужик (Щедин?) доказал, что можно программировать без присваивания. Так как присваивание нарушает законы математической логики.

Затем появились функциональные языки. И, допустим, в них нужно заменять цикл рекурсией, т.к. у них нет переменной-счетчика i.

Монада (уточнить, что это).

Процедурные языки —

Темпоральные логики (уточнить, что это такое).

ТРЭПЛО — теоретические разработки эвристического поиска логического обоснования.

6 Виртуальные машины. Понятие виртуальной машины. Иерархия ВМ. Промежуточные языки.

Система команд машины. (те самые +, -, if, while и так далее). Каждая команда реализуется в виде нескольких т.н. микрокоманд (микропрограмм). Микропрограммы опираются на устройства: АЛУ, память, система управления прерываниями — т.н. блоки ЭВМ. Из блоков составляются логические схемы при помощи вентилей. И самой мелкой составляющей является триггер.

Сейчас все пишут на алгоритмических языках высокого уровня, которые работают на ОС, которая, в свою очередь, работает над базами данных и так далее. Каждый уровень такой системы называется виртуальной машиной.

Т.о. ЭВМ — многоуровневое устройство, каждый уровень которой называется виртуальной машиной.

Никлас Вирт — собственно, автор первой виртуальной машины, которую он ПРИДУМАЛ (то есть она существует в его чертогах разума), которая имела всего 20 команд, называлась она VM (Pcode). Он сделал транслятор Pascal под VM. А потом переписал этот транслятор на Pascal.

Преимущества виртуальной машины — переносимость (главная характеристика), универсальность, легкость понимания.

Был язык FORTH, в котором был сделан такой финт ушами: шитый код.

Примеры для экзаменов: ЕС-63 (интерпретатор в дисплее) и перефирийное устройство управления ФОБОС-К.

Вы понимаете, что когда вы пишете программу, вы пишете программу для какой-либо виртуальной машины? Нет? Вы в дерьме :) C++ — трансляторы, Asm — минуя транслятор, но и здесь не обращение напрямую к железу.

Виртуальная машина может читать программу побайтно и переводить её в язык машинных кодов. Такая виртуальная машина называется интерпретатор.

Кен Олсон — придурок из компании DEC, который не обратил внимания на рынок персональных компьютеров.

7 Введение в теорию трансляции: сравнение интерпретаторов и компиляторов. Стадии трансляции. Машинно-зависимая и машина-независимая части транслятора.

Есть программа на АЯВУ (алгоритмический язык высокого уровня). А мы хотим получить код для x86_64. Происходят следующие этапы:

1. Сначала работает лексер (сканер). Он занимается свертками (т.е. сворачивает лексеммы типа 3.14, abc в float-число 3.14 и переменную).

2. Теперь работает парсер (синтаксический видо-независимый анализатор). Он «разбирает» сложные команды наподобие a[3]+=abc+1 в дерево разбора.

3. Видозависимый анализ. Он должен понять, что если стоит 1+3.14, то слева целый операнд, а справа вещественный, то есть результат будет вещественным.

4. Оптимизация, которая делится на 2 части: машинно-независимая (создание рабочей ячейки с данными, которые могут быть использованы как результат частовыполняющихся одних и тех же команд) и машинно-зависимая (смотри пример внизу).

5. Генерация кода — на выходе получается объектный код (.out-файлы).

6. Затем работает линковщик, который связывает несколько файлов объектного кода и собирает их в одну программу.

7. И под конец шлифует все некий «лодырь».

```
for (int i = 0; i < n; i++)
{
    a[i] = b[i]+c[i];
}
```

Пример машинно-зависимой оптимизации

Индуцированная переменная — для каждого из трех массивов заводим переменную (рабочую ячейку) и в конце цикла делать наращивание на шаг (допустим, 4 байта).

То, что получается после каждого шага — это промежуточный код.

Транслятор — это либо компилятор, либо интерпретатор. Этапы, записанные выше, — это этапы компиляции. Выполнение компилированной программы — подача на вход загрузочного модуля и данных, и получение результата. А интерпретатор — это подача данных и каждой новой строчки кода, которая переводится в машинный код отдельно.

Главное назначение интерпретаторов — отладка.

В 1965 году академик Глушков в институте кибернетики сделал машину МИР-1, один из первых HLL. Ему на вход подавался текст, а он его исполнял построчно.

8 Введение в СУБД. История и причины возникновения систем баз данных, использование языков запросов баз данных.

СУБД — система управления базами данных.

Сначала были сетевые базы данных.

Затем иерархические (деревянные) базы данных.

И наконец некто Кодд придумал реляционную алгебру и на её основе были разработаны реляционные базы данных.

Реляционная база данных — это таблица прямоугольная:

	$k1$	$k2$	$k3$	$k4$	$k5$	$k6$	$k7$	$k8$	$k9$	$k10$
0										
1										
2										
3										
4	з	н	а	ч	е	н	и	я		
5										
6										
7										
8										
9										

Пишется все это на языке SQL (декларативный язык).

```
select a,b,c From T1, T2 // где a,b,c --- атрибуты и T1, T2 --- таблицы.
where () // условие выбора
```

База называется нормализованной, если ключи не повторяются (по ключу однозначно можно выбрать строки).

Главная БД — Oracle. Затем более-менее хорошая MS SQL. Еще имеется открытая база данных MySQL.

Еще несколько: Postgress, PostgreeSQL, DB2;

9 Эволюция программ: сопровождение, характеристики удобного для сопровождения ПО, реинжиниринг, унаследованные системы, повторное использование ПО.

Maintenance — сопровождение. Если взять деньги, отпущенные на программный продукт, то 30% из них уйдут на разработку, а 70% — на это самое сопровождение.

Сопровождение — это:

1) Дебаггинг и поддержка пользователей.

2) Улучшение и оптимизация программы.

3) Адаптирование под новые технологии.

4) Улучшение UX — «удобство использования программы».

5) Всемерное улучшение документации.

Теперь вместо Maintenance говорят Evolution (модификация терминов у них такая).