Основы алгоритмизации

Языки программирования

Проверочная работа

1 вариант

1. Напишите свойства алгоритмов
2. Приведите виды разветвляющихся алгоритмических структур (в виде блок-схем)
3. Найти сумму первых n степеней двойки, начиная с нулевой степени: $1 + 2 + 4 + \ldots + 2^{n-1}$.

2 вариант

1. Дайте определение понятию «алгоритм»
2. Приведите виды циклических алгоритмических структур (в виде блок-схем)
3. Услуги телефонной сети оплачиваются по следующему правилу: за разговоры до *A* минут в месяц оплачиваются *B* р., а разговоры сверх установленной нормы оплачиваются из расчета *C* р. в минуту. Написать алгоритм, вычисляющий плату за пользование телефоном для введенного времени разговоров за месяц.

Языки программирования

Язык программирования содержит три основных компонента: ***алфавит***, ***синтаксис*** и ***семантику***. Эти компоненты определяют правила записи программ.

Алфавит языка – это фиксированный набор символов, которые можно применять в инструкциях языка программирования. Другие символы допустимы только в особых случаях, например в строковых константах.

Из символов алфавита формируются лексемы языка:

- константы;
- идентификаторы;
- знаки операций;
- ключевые (служебные, иначе зарезервированные) слова;
- разделители (знаки пунктуации).

Синтаксис языка определяет правила построения операторов.

Семантика – это смысловое содержание операторов языка программирования. Семантические правила определяют действия, описываемые различными операторами, и, в итоге сущность всего алгоритма.

Языки программирования

Язык программирования содержит три основных компонента: ***алфавит***, ***синтаксис*** и ***семантику***. Эти компоненты определяют правила записи программ.

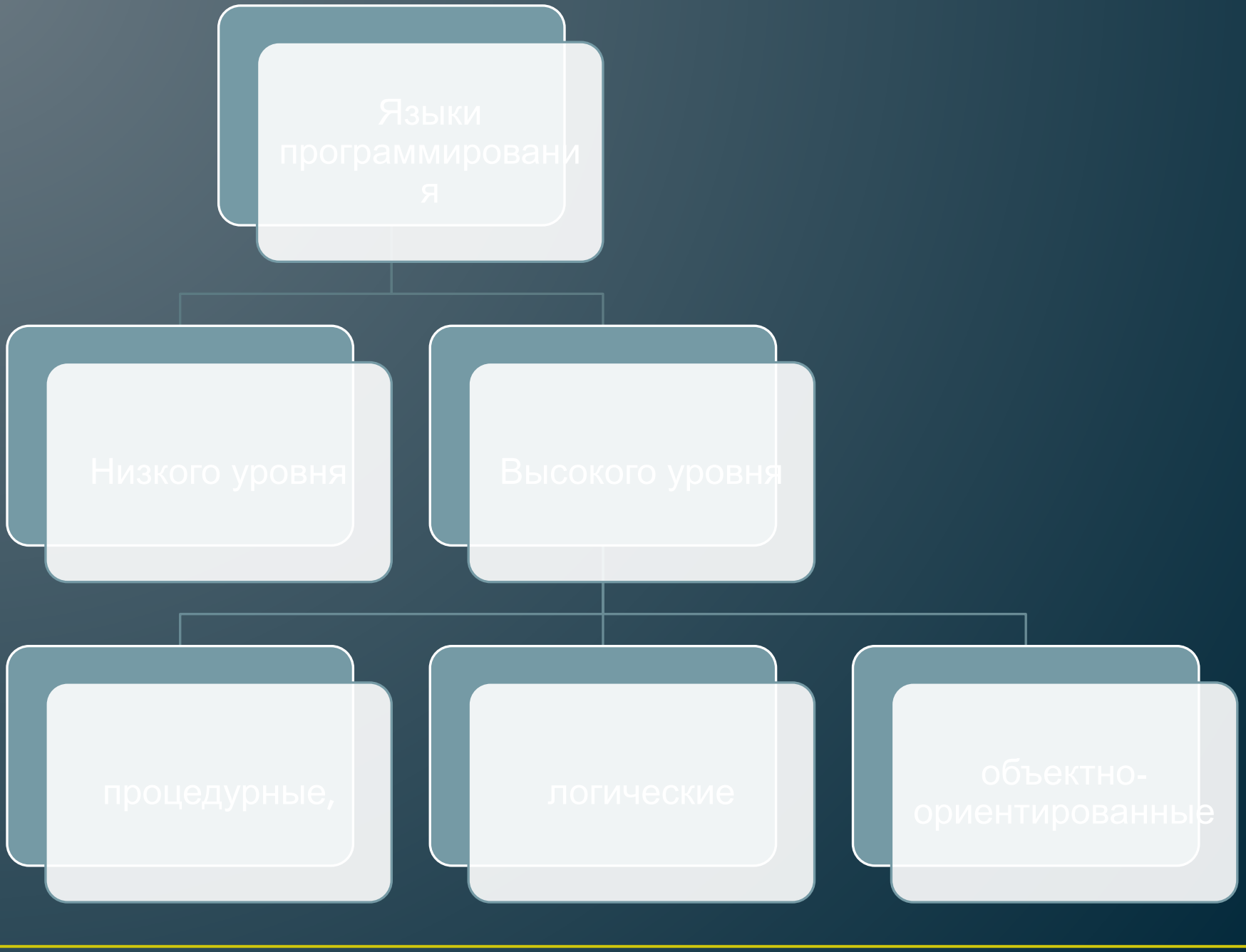
Алфавит языка – это набор символов, которые можно применять в инструкциях языка программирования. Другие символы допустимы только в особых случаях, например в строковых константах.

Синтаксис языка определяет правила построения операторов.

Семантика – это смысловое содержание операторов языка программирования. Семантические правила определяют действия, описываемые различными операторами, и, в итоге сущность всего алгоритма.

Языки программирования

Языки низкого уровня	Языки высокого уровня
К языкам низкого уровня относится язык Ассемблер, в котором программа пишется на уровне машинных кодов. Инструкция языка ассемблера описывает ровно одну машинную команду. И наоборот: каждой команде в системе команд процессора соответствует инструкция языка (мнемоника). По сравнению с машинным кодом язык ассемблер имеет ряд преимуществ, облегчающих труд программиста: • Символические мнемоники запоминаются легче, чем шестнадцатеричные коды команд. • Для регистров и областей памяти также можно использовать символические имена. • Нет необходимости работать с физическими адресами памяти. • Числовые константы и строки представляются в программе в привычном виде.	Другие языки программирования - высокого уровня, в которых каждая инструкция (оператор) языка преобразуется в группу машинных. Эти языки ориентируются не на систему команд процессора, а на способ мышления, присущий человеку. Достоинства языков высокого уровня: • Машинная независимость. • Использование естественных обозначений. • Эффективное представление этапов обработки данных средствами языка. • Готовые библиотеки стандартных подпрограмм для выполнения часто встречающихся действий.


Языки программирования
Низкого уровня
Высокого уровня
процедурные,
логические
объектно-ориентированные

Поколения языков: 1 этап

Операционное программирование. (ЭВМ 1-го поколения с 1945-1959 год). ЭВМ того времени понимали только цифровые команды, и программы состояли из множества строк, состоящих из цифр, интерпретируемых центральным процессором. Например, команда 05 825 631 трактовалась как сложение двух чисел (код 05), записанных в ячейки с номерами 825 и 631.

Производительность труда программистов того времени была очень невелика, так как вручную было необходимо распределить все переменные программы в оперативной памяти.

Поколения языков: 2 этап

Он связан с ЭВМ 2-го поколения. Появились языки программирования типа Ассемблер и автокод. Теперь команда сложения записывалась с использованием служебных слов – ADD (сложить) PR1, ZET, где ADD – код команды, PR1, ZET – имена ячеек.

Перевод программы (трансляция), записанных таким образом в цифровое представление, а только такое понимает ЭВМ, осуществляется с помощью специальных программ, называемых ассемблерами.

Технология работы программиста: программа собирается из мелких деталей, отдельных операций и имеет достаточно простую структуру, решаемые задачи в основном расчетные.

Поколения языков: 3 этап

Развиваются языки программирования высокого уровня. В них реализуются новые идеи: подпрограммы и раздельная компиляция (Фортран 2); блочная структура и типы данных (Алгол 60); описание данных и работа с файлами (Кобол); обработка списков и указателей (Лисп). В следующих версиях языков продолжается развитие: PL/1 (Фортран+Алгол+Кобол), Алгол 68 (приемник Алгол 60), Паскаль (развитие Алгол 60), Simula (классы абстрактные данные).

Возможности языков программирования обеспечивают поддержку нисходящей технологии конструирования программ. Суть нисходящего конструирования программ – разбиение большой задачи на подзадачи, которые могут рассматриваться отдельно.

Поколения языков: 3 этап

Основные правила применения данной технологии:

- формализованное и строгое описание программистом входов функций и выходов всех модулей программы и системы;
- согласованная разработка структур данных и алгоритмов;
- ограничение на размер модулей.

Восходящая технология конструирования программ – решение складывается «из отдельных кирпичиков», из известных решений подзадач. Таким образом, данной технологией оговаривается определенный принцип декомпозиции и иерархическая структура программы. Важнейшей составляющей этой технологии является ***структурное программирование*** (языки программирования Паскаль, Модула–2). Пионером структурного программирования был Э. Дейкстра, который в 1965 году предположил, что оператор GOTO может быть

Поколения языков: 3 этап

Характерные черты структурного стиля программирования:

- простота и ясность (программа легко читается и анализируется);
- использование только базовых конструкций;
- отсутствие сетевых структур в программе;
- отсутствие многоцелевых функциональных блоков;
- отсутствие неоправданно сложных арифметических и логических конструкций;
- расположение в строке программы не более одного оператора языка программирования;
- содержательность имен переменных.

При этом процесс нисходящей разработки программы может продолжаться до тех пор, пока не будет, достигнут уровень «атомарных» блоков, т.е. базовых конструкций.

Поколения языков: 3 этап

К нисходящей технологии следует отнести и то, что называется ***модульным программированием***.

Достаточно независимые фрагменты задачи оформляются как ***модули***. Создаются *библиотеки модулей*, определяется механизм включения модулей в разрабатываемую программу.

Модуль должен иметь строго определенный интерфейс и скрытую часть, одну точку входа и одну точку выхода.

Структурная технология предоставила в распоряжение разработчиков строгие, формализованные методы описания программ и принимаемых технических решений. При этом использовалась наглядная графическая техника (схемы, диаграммы). Программы имели последовательную структуру, идеи Э. Дейкстры были реализованы в полной мере, что определило новый этап в развитии технологии программирования.

Поколения языков: 4 этап

Связан с применением объектно-ориентированных языков 4-го поколения.

В основе объектно-ориентированного программирования (ООП) лежит идея объединения в одной структуре данных и действий, которые производятся с этими данными. В терминологии ООП такие действия называются ***методами***.

При таком подходе организация данных и программная реализация действий над ними оказываются гораздо сильнее связаны, чем при традиционном структурном программировании.

Поколения языков: 4 этап

ООП базируется на трех основных понятиях:

Инкапсуляция – комбинирование данных с процедурами и функциями, которые манипулируют этими данными. В результате получается новый тип данных – объект.

Наследование – это возможность использования уже определенных объектов для построения иерархии объектов производных от них. Каждый из «наследников» наследует описание данных «прародителя» и доступ к методам их обработки.

Полиморфизм – это возможность определения единого по имени действия (процедуры или функции), применимого одновременно ко всем объектам иерархии наследования, причем каждый объект иерархии может «заказывать» особенность реализации этого действия над «самим собой».

Поколения языков: 4 этап

ООП может заметно упростить написание сложных программ, придать им гибкость. Одним из его главных преимуществ можно назвать возможность расширять область их применения, не переделывая программу, а лишь добавляя в нее новые уровни иерархии.

Первым языком с элементами ООП был язык Симула-67. В Турбо-Паскале, начиная с версии 5.5, появились средства ООП. Итогом развития Турбо-Паскаля в этом направлении стало создание фирмой Borland системы программирования Delphi (Делфи). Использование этой системы, в частности, даёт возможность легко и быстро программировать сложный графический интерфейс.

Области применения языков программирования

В настоящее время языки программирования применяются в самых различных областях человеческой деятельности, таких как:

- научные вычисления (языки C++, FORTRAN, Java);
- системное программирование (языки C++, Java);
- обработка информации (языки C++, COBOL, Java);
- искусственный интеллект (LISP, Prolog);
- издательская деятельность (Postscript, TeX);
- удаленная обработка информации (Perl, PHP, Java, C++);
- описание документов (HTML, XML).

С течением времени одни языки развивались, приобретали новые черты и остались востребованы, другие утратили свою актуальность и сегодня представляют в лучшем случае чисто теоретический интерес. В значительной степени это связано с такими факторами, как:

- наличие среды программирования, поддерживающей разработку приложений на конкретном языке программирования;
- *удобство сопровождения* и тестирования программ;
- стоимость разработки с применением конкретного языка программирования;
- четкость и ортогональность конструкций языка;
- применение объектно-ориентированного подхода.

Парадигмы программирования

Синтаксис языка описывает систему правил написания различных языковых конструкций, а *семантика* языка программирования определяет смысл этих конструкций.

Семантика языка взаимосвязана с используемой *вычислительной моделью*. В настоящее время языки программирования в зависимости от применяемой *вычислительной модели* делятся на четыре основные группы:

- **Процедурные языки**
- **Аппликативные языки**
- **Языки системы правил**
- **Объектно-ориентированные языки**

Процедурные языки

представляют собой ***последовательность выполняемых операторов***. Если рассматривать состояние ПК как состояние ячеек памяти, то процедурный язык – это последовательность операторов, изменяющих значение одной или нескольких ячеек.

К процедурным языкам относятся FORTRAN, C, Ada, Pascal, Smalltalk и некоторые другие.

Процедурные языки иногда также называются ***императивными языками***.

Код программы на процедурном языке может быть записан следующим образом:

- Operator1; operator2; operator3

Аппликативные языки

В основу данного вида языков положен функциональный подход.

Язык рассматривается с точки зрения нахождения функции, необходимой для перевода памяти ПК из одного состояния в другое.

Программа представляет собой набор функций, применяемых к начальным данным, позволяющий получить требуемый результат.

К *аппликативным языкам* относится язык LISP.

Код программы на *аппликативном языке* может быть записан следующим образом:

- function1(function2(function3(beginning_date)))

Языки системы правил

Называемые также ***языками логического программирования***, основываются на определении набора правил, при выполнении которых возможно выполнение определенных действий.

Правила могут задаваться в виде утверждений и в виде таблиц решений.

К языкам логического программирования относится язык Prolog.

Код программы на языке системы правил может быть записан следующим образом:

```
if condition1 then operator1;
if condition2 then operator2;
if condition3 then operator3
```

Объектно-ориентированные языки

основанные на построении объектов как набора данных и операций над ними.

Объектно-ориентированные языки объединяют и расширяют возможности, присущие *процедурным* и *аппликативным языкам*.

К объектно-ориентированным языкам относятся C++, *Object Pascal*, Java.

В настоящий момент наибольшее распространение получили языки, основанные на объектно-ориентированной модели. Они, реализуя *процедурную модель* построения языка, поддерживают аппликативность конструкций, позволяя представлять блок-схему выполнения *структурированной программы* как некоторый набор аппликативных функций.

Стандартизация языков программирования

Концепция языка программирования неотрывно связана с его реализацией.

Для того чтобы *компиляция* одной и той же программы различными компиляторами всегда давала одинаковый результат, разрабатываются стандарты языков программирования.

Существует ряд организаций, целенаправленно занимающихся вопросами стандартизации. Например,

- Американский национальный институт стандартов *ANSI* (American National Standards Institute),
- Институт инженеров по электротехнике и электронике *IEEE* (Institute of *Electrical and Electronic Engineers*),
- Организация международных стандартов *ISO* (*International* Organization for *Standardization*).

Стандартизация языков программирования

Как правило, при создании языка выпускается частный стандарт, определяемый разработчиками языка. Если язык получает широкое распространение, то со временем появляются различные версии компиляторов, которые не точно следуют частному стандарту. В большинстве случаев идет расширение зафиксированных первоначально возможностей языка. Для приведения наиболее популярных реализаций языка в соответствие друг с другом разрабатывается согласительный стандарт. Очень важным фактором стандартизации языка программирования является своевременность появления стандарта – до широкого распространения языка и создания *множества* несовместимых реализаций. В процессе развития языка могут появляться новые стандарты, отражающие современные нововведения.

Стандартизация языков программирования

В процессе развития языка некоторые его конструкции и функции устаревают. Однако с целью обратной совместимости новые версии должны поддерживать и все устаревающие возможности. Это ведет к "разбуханию" компиляторов.

В последнее время в реализациях введено понятие ***не рекомендуемой*** и ***устаревшей возможности***.

В первом случае следующий стандарт еще будет поддерживать не рекомендуемую возможность, но может перевести ее в категорию устаревшей.

Во втором случае стандарт может исключить поддержку возможности, объявленной ранее как устаревшая.

Введение не рекомендуемых и устаревших возможностей предоставляет разработчикам временной интервал, в течение которого они могут модифицировать код в соответствии с новыми требованиями стандарта.

Среда проектирования

С развитием языков программирования совершенствовались и *средства разработки* программ – от режима командной строки до интегрированной среды проектирования. Такая среда предоставляет удобный графический *интерфейс* разработки и большой спектр сервисов, включающих управление версиями хранимых данных, утилиты просмотра и управления информацией, библиотеки классов, мастера создания шаблонов приложений и т.п.

Компилятор языка программирования выступает как составная часть среды проектирования. Сама *программа* наряду с конструкциями, предусмотренными стандартом, как правило, использует библиотечные функции и классы, предоставляемые средой проектирования.

Так, интегрированная *среда разработки* VisualStudio.*NET* содержит библиотеку классов *MFC* (Microsoft *Foundation* Classes), значительно упрощающую процесс разработки приложений, использующих оконный *интерфейс*.

Среда проектирования

Интегрированная среда проектирования VisualStudio.*NET* позволяет создавать и компилировать приложения на языках C++, C#, *Visual Basic* и VisualJ. Для разработки приложений на языке C++ предназначается также среда CBuilder.

Для проектирования приложений на языке *Object Pascal* используется интегрированная среда проектирования Delphi.

Наиболее удобной средой разработки программ на языке *Java* является интегрированная среда проектирования JBuilder.