*Лекции по курсу «Языки программирования»

2018г. КИТУС

Преподаватель Исаева Г.Н.

Языки программирования высокого уровня

Этапы решения задачи на ЭВМ

1. Постановка задачи:

* • сбор информации о задаче;
* • формулировка условия задачи;
* • определение конечных целей решения задачи;
* • определение формы выдачи результатов;
* • описание данных (их типов, диапазонов величин, структуры и т. п.).

Этапы решения задачи на эвм

2. Анализ и исследование задачи, модели:

* • анализ существующих аналогов;
* • анализ технических и программных средств;
* • разработка математической модели;
* • разработка структур данных.

3. Разработка алгоритма:

- выбор метода проектирования алгоритма;
- выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
- выбор тестов и метода тестирования;
- проектирование алгоритма.

4. Программирование:

- выбор языка программирования;
- уточнение способов организации данных;
- запись алгоритма на выбранном языке
- программирования.

5. Тестирование и отладка:

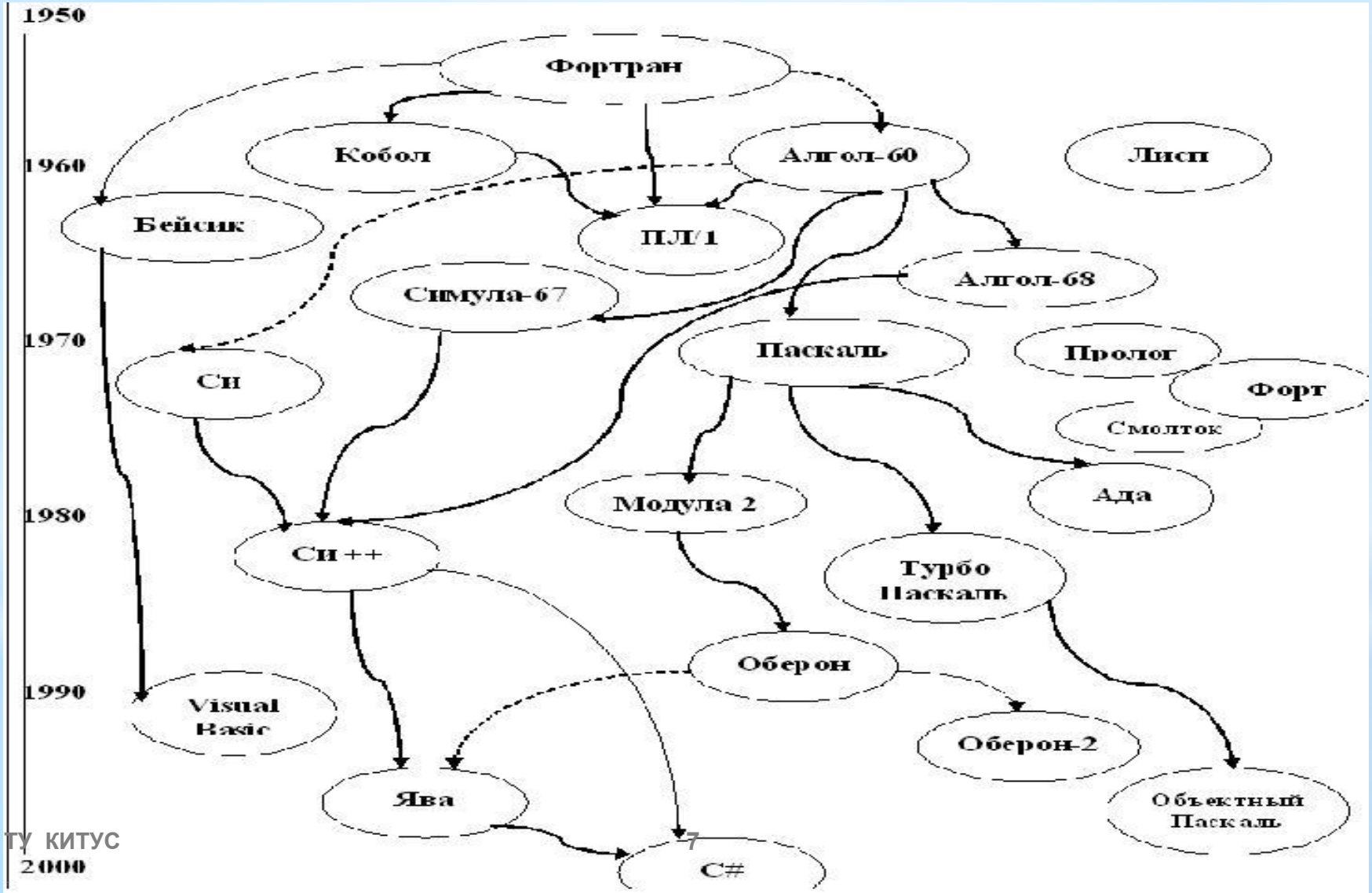
* • синтаксическая отладка;
* • отладка семантики и логической структуры;
* • тестовые расчеты и анализ результатов тестирования;
* • совершенствование программы.

6. Анализ результатов решения задачи и уточнение в случае необходимости математической модели с повторным выполнением этапов 2-5.

7. Сопровождение программы:

* • доработка программы для решения конкретных задач;
* • составление документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию.

Генеалогия языков программирования высокого уровня



Язык программирования

- формальная знаковая система, предназначенная для записи программ.
- Язык программирования определяет набор лексических, синтаксических и семантических правил, используемых при составлении компьютерной программы.
- ЯП позволяет программисту точно определить то, на какие события будет реагировать компьютер, как будут храниться и передаваться данные, а также какие именно действия следует выполнять над этими данными при различных обстоятельствах.

*Язык программирования

Со времени создания первых программируемых машин человечество придумало уже более восьми с половиной тысяч языков программирования. Каждый год их число пополняется новыми.

*язык программирования

язык программирования отличается от естественных языков тем, что предназначен для передачи команд и данных от человека компьютеру, в то время, как естественные языки используются для общения людей между собой.

Нотации

Для описания синтаксических конструкций языков программирования используются две нотации:

- Бэкуса (впервые предложена при описании языка ALGOL);
- IBM (разработана фирмой для описания языков COBOL и JCL).

Нотация Бэкуса содержит конструкции следующего вида:

<Оператор присваивания> ::= <Переменная> := <Выражение>

<Слово> ::= <Буква>|<Слово><Буква>

Левая часть определения конструкции языка содержит наименование определяемого элемента, взятого в угловые скобки.

Правая часть включает совокупность элементов, соединенных знаком | , который трактуется как «или» и объединяет альтернативы — различные варианты значения определяемого элемента.

Части соединяются оператором ::=, который означает *есть по определению*.

Нотация IBM включает следующие конструкции:

< > — *угловые скобки (или двойные кавычки " ") обозначают* элементы программы, определяемые пользователем

[ ] — *квадратные скобки, ограничивающие синтаксическую* конструкцию, обозначают ее возможное отсутствие.

Например:

return [<выражение>]; В этой конструкции <выражение> не обязательно;

| — вертикальная черта разделяет список значений обязательных элементов, одно из которых должно быть выбрано;

... — *горизонтальное многоточие, следующее после некоторой* синтаксической конструкции, обозначает последовательность конструкций той же самой формы, что и предшествующая многоточию конструкция.

Данные и типы данных

Данные — это любая информа*ция*, представленная в формализованном виде и пригодная для обработки алгоритмом.

Данные делятся на переменные и константы.

Переменные — это такие данные, значения которых могут изменяться в процессе выполнения алгоритма.

Константы — это данные, значения которых не меняются в процессе выполнения алгоритма.

Каждая переменная и константа должна иметь свое уникальное *имя*. *Имена переменных и констант задаются идентификаторами*.

Идентификатор (по определению) представляет собой последовательность букв и цифр.

Типичные группы *функций:*

- **стандартные** алгебраические и арифметические — SIN, COS, SQRT, M IN , MAX и др.;
- **стандартные строчные** — выделение, удаление подстроки, проверка типа переменной и т. д.;
- **нестандартные функции**, в том числе: описание операций и форматов ввода-вывода данных; преобразование типов данных; описание операций над данными, специфичными для конкретной системы программирования, ОС или типа ЭВМ.

Данные и типы данных

Тип данных — это такая характеристика данных, которая, с одной стороны, задает *множество значений для возможного изменения* данных и, с другой стороны, определяет *множество операций, которые можно к этим данным применять, и правила* выполнения этих операций.

Язык программирования

Создан под влиянием АЛГОЛ, Pascal

Ada — мощнейший модульный объектно-ориентированный язык общего назначения, ориентированный на разработку надёжного программного обеспечения.

Ada был создан в 1979–1980 годах па заказу Министерства Обороны США. Целью проекта было уменьшение количества различных языков, используемых в Министерстве для различных целей (на 1983 год — свыше 450) путем разработки единого языка, удовлетворяющего требованиям Министерства.

*Ada

Парадигма:

* императивная
* мультипарадигма
* на уровне значений
* обобщённая
* объектно-ориентированная
* процедурная
* скалярная
* строгая
* структурная

* **Ada**

*Ada

Диалекты:

Ada 2005
Ada 83
Ada 95

Реализации и версии

A#
Ada/Ed
Green Hills
Hewlett-Packard Ada
IBM Rational Ada Developer
Sun SC Ada
gnat
    gnat 3.4.5
    Принятые расширения файлов:.adb, .ads, .ada

*Ada

Hello, World!:

(Пример для версий gnat .4.5)

```
with Ada.Text_IO;
 procedure HelloWorld is
begin
Ada.Text_IO.Put_Line("Hello, World!");
end HelloWorld;
```

Модульное программирование

В Паскале задача может быть разделена на более простые и понятные фрагменты — подпрограммы, после чего программу можно рассматривать в более укрупненном виде — на уровне взаимодействия подпрограмм.

Модульное программирование

Следующим шагом в повышении уровня абстракции программы является группировка подпрограмм и связанных с ними данных в отдельные файлы (**модули**), компилируемые раздельно.

Модульное программирование

Разбиение на модули уменьшает время перекомпиляции и облегчает процесс отладки, скрывая несущественные детали за интерфейсом модуля и позволяя отлаживать программу по частям (при этом, возможно, разными программистами).

Подпрограммы

* Подпрограмма — это фрагмент кода, к которому можно обратиться по имени
* Логические законченные части программы оформляются в виде подпрограмм
* Подпрограмма записывается один раз, а вызываться может столько раз, сколько необходимо
* Одна и та же подпрограмма может обрабатывать различные данные, переданные ей в качестве параметров.

Подпрограммы

Объявления (типы, переменные, константы), использующиеся любой подпрограммой, относятся к одной из двух категорий:

локальных объявлений и ***глобальных***

объявлений.

Локальные объявления принадлежат подпрограмме, описаны внутри нее и могут использоваться только ею.

Переменные, объявленные таким образом, формируются автоматически при передаче управления процедуре и уничтожаются при выходе из нее.

Подпрограммы

Глобальные объявления принадлежат программе в целом и доступны как самой программе, так и всем ее подпрограммам.

Обмен данными между основной программой и ее подпрограммами обычно осуществляется посредством глобальных переменных.

Если имя глобального объявления совпадает с именем локального, то внутри подпрограммы объявление обычно интерпретируется как локальное, и все изменения, вносимые, например, в значение такой переменной, актуальны только *в рамках подпрограммы*.

Параметры подпрограмм

Подпрограмма выполняет преобразование входных параметров в выходные — это есть отображение набора значений входных

аргументов в выходной набор результатов с возможной модификацией входных значений.

Объявление подпрограммы может содержать список параметров,

которые называются ***формальными***

Формальные и фактические параметры

Каждый параметр из списка формальных параметров является локальным по отношению к подпрограмме, для которой он объявлен. Это означает, что глобальные переменные, имена которых совпадают с именами формальных параметров, становятся недоступными для использования в подпрограмме.

Формальные и фактические параметры

формальные параметры можно разбить на две категории:

• параметры, вызываемые подпрограммой ***по своему значению***(т. е. параметры, которые передают в подпрограмму свое значение и не меняются в результате выполнения подпрограммы);

• параметры, вызываемые подпрограммой ***по наименованию*** (т. е. параметры, которые становятся доступными для изменения внутри подпрограммы).

Формальные и фактические параметры

Главное различие этих двух категорий — в механизме передачи параметров в подпрограмму.

При обращении к подпрограмме формальные параметры заменяются на соответствующие по типу и категории ***фактические***

параметры вызывающей программы или подпрограммы.

Структура программы на языке высокого уровня

1. раздела идентификации – области, содержащей наименование программы, а также дополнительную информацию для программистов и/или пользователей;
2. раздела связи – фрагмента текста, описывающего внешние переменные, передаваемые вызывающей программой (если таковая имеется) Эти переменные часто называют **параметрами программы**;

Структура программы на языке высокого уровня

3. раздела оборудования (среда) — описания типа ЭВМ, процессорацессора,требований к оперативной и внешней памяти, существенных

с точки зрения выполнимости программы;

4. раздела данных — идентификации (декларация, объявление, описание) переменных, используемых в программе, и их типов.

Структура программы на языке высокого уровня

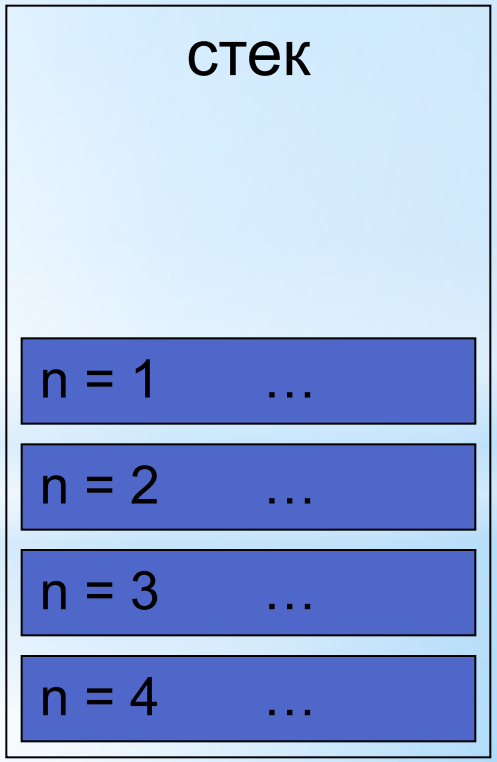
5. раздела процедур — собственно программной части, содержащей описание процессов обработки данных.

Элементами

процедуры являются операторы и стандартные функции,входящие в состав соответствующего языка программирования.

*Рекурсивные подпрограммы

```
function FTR(n : byte): longint;
begin
      if (n = 0) or (n = 1) then FTR := 1
      else   FTR := FTR(n - 1) * n;
end;
----------
A := FTR(4);
```



Обработка исключений

В языке *C++* существует инструмент, который называется *обработкой исключительных ситуаций.* Этот инструмент предоставляет программе возможность, в случае возникновения ошибки, не просто прекратить выполнение, а как-то обработать эту ошибку (исключительную ситуацию).

Обработка исключений

Для обработки исключительных ситуаций в языке *C++* используются

три оператора *try*, *catch* и *throw*. Коды директив, которые применяются для

контроля возникновения ошибки, располагают в блоке операторов за ключевым словом *try*.

Блоки

try { <директивы проверок ошибок> }

надо размещать в программе там, где возможно появление ошибки.

Обработка исключений

После выявление ошибки управление передается блоку операторов, который следует за ключевым словом *catch*.

Блок обработки *catch* (<тип> <имя>)

{<директивы обработки ошибки> } располагаются непосредственно после блока *try*.

Блок *catch* классифицирует

выявленные исключения.

Обработка исключений

Общий формат обработки исключений выглядит следующим образом:

```
try
{
<Операторы, проверяющие наличие ошибки.>
}catch( <тип1> <имя1>)
{
}
catch( <тип2> <имя2>){
}
...
```

Обработка исключений

Инструкция *throw* находится внутри блока *try* (или внутри функций,

которые вызывает блок *try*), и она передает управление обработчику, то есть

блоку *catch*. …