Лекция 5.

Технология разработки программного обеспечения

Жизненный цикл программы состоит из этапа разработки программы и этапа её эксплуатации и сопровождения (контроль работоспособности и при необходимости внесение изменений и дополнений).

Технология разработки ПО – совокупность приёмов, позволяющих создать безошибочную программу в течение заданного времени. Состоит из четырёх этапов:

1) формулировка задачи на естественном языке и создание математической модели;
2) разработка нового или выбор существующего метода численного решения математической задачи (алгоритма);
3) написание программы на языке программирования;
4) тестирование и отладка программ.

На первом этапе необходимо наиболее глубоко исследовать предметную область (процесс, объект, явление), а также разработать наиболее полную математическую модель, учитывающую основные особенности предметной области.

На втором этапе при разработке алгоритма необходимо использовать приёмы *структурного программирования* (см. ниже), позволяющие создавать надёжно работающие программы. Алгоритм принято представлять в виде графической схемы, которая составляется из нескольких геометрических фигур – блоков. Основные блоки схемы алгоритма выглядят следующим образом (рисунок 9):

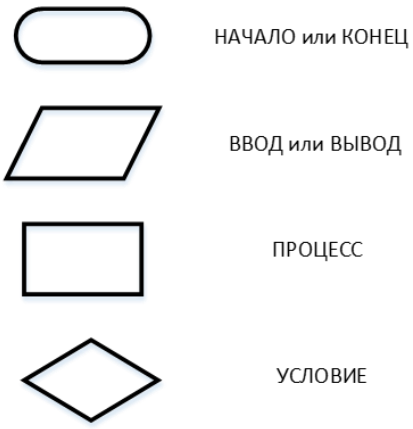

Рисунок 9 – Блоки схемы алгоритма

Например, схема алгоритма простейшей программы линейной структуры (ввод, сложение двух чисел A, B и вывод результата C) выглядит следующим образом (рисунок 10):

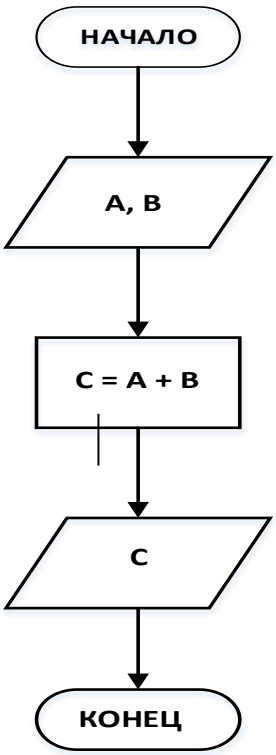

Рисунок 10 – Схема алгоритма линейной структуры

А схема оператора условной передачи управления (if A then B else C, где A – условие, B – действие, выполняющееся при истинности A, а C – действие, выполняющееся в противном случае) выглядит так (рисунок 11):

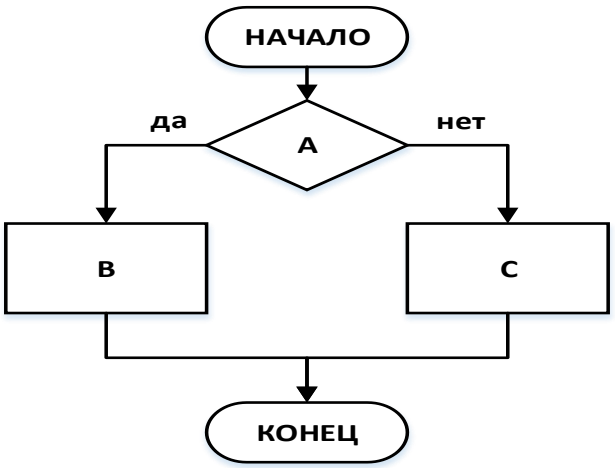

Рисунок 11 – Схема условного оператора

Для оператора цикла с известным числом повторений (for I:=N to M do S) схема

выглядит следующим образом (рисунок 12):

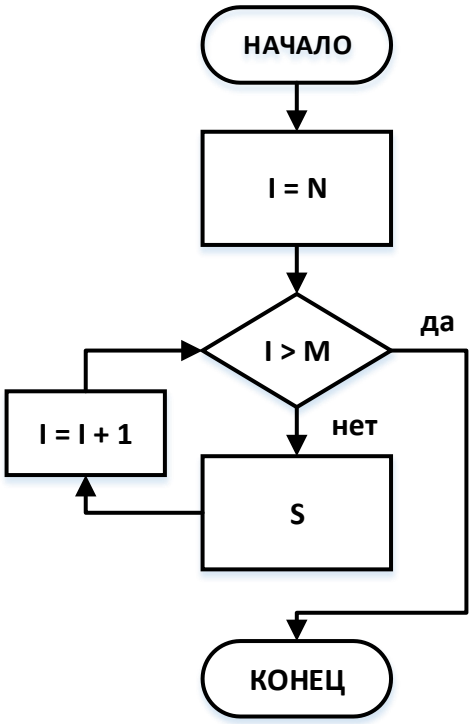

Рисунок 12 – Схема оператора цикла с известным числом повторений

Для операторов цикла с неизвестным числом повторений с предусловием (while A do S) и постусловием (repeat S until B) схемы выглядит следующим образом (рисунок 13):

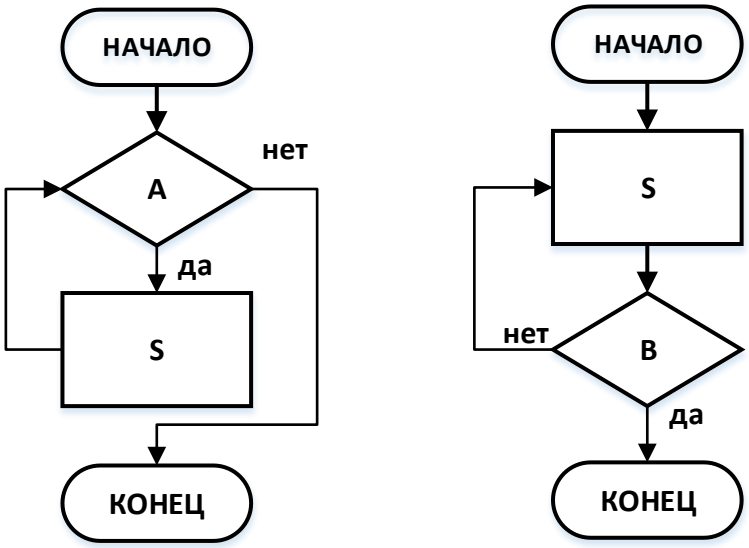

Рисунок 13 – Схемы операторов цикла с неизвестным числом повторений

На третьем этапе при выборе языка программирования необходимо учитывать тип решаемой задачи. Например, для вычислительных задач удобнее использовать язык С,

Fortran и подобные им. При разработке интернет-приложений – язык Java. Языки Pascal, Basic считаются универсальными и часто используются для обучения программированию.

Для создания безошибочной программы за приемлемое время используются основные приемы *структурного программирования.*

Суть его заключается в следующем:

Исходная сложная задача условно разбивается на более простые подзадачи, которые являются относительно независимыми друг от друга. Каждая из этих задач программируются в отдельной программе-модуле. Эти прикладные модули объединяются в единое целое специальным управляющим модулем, который может входить в группу подобных модулей (в случае решения сложных задач), объединённых основным управляющим модулем. В результате получается структурированная иерархическая система – программа, представляющая собой композицию из последовательных или вложенных друг в друга модулей.

Принципы разбиения на подзадачи-модули:

- доступность восприятия;
- незначительный размер (желательно не более 100 строк программы);
- учет возможностей изменения модуля в дальнейшем;
- учет наличия уже готовых модулей.

Модульный подход имеет положительные стороны:

- упрощение создания и дальнейшей модификации программ;
- создание библиотеки модулей;
- возможность параллельной работы с несколькими модулями одновременно;
- уменьшение объема, занимаемой ОП компьютера.

На уровне прикладных модулей при программировании используются три основные *базовые (управляющие) конструкции*, которые могут изменять ход вычислительного процесса:

- *Конструкция следования* (например, оператор GOTO);
- *Конструкции ветвления*:
  - конструкция условного ветвления (IF);
  - конструкция выбора (CASE).

Обе эти конструкции могут быть полные и неполные (без ELSE).

- *Конструкции повторения*:
  - с известным числом повторений (FOR);
  - с неизвестным числом повторений:
    - ✓ с предусловие (WHILE);
    - ✓ с постусловие (REPEAT).

Существует *два метода создания многомодульных пакетов программ*:

1. *Метод восходящего проектирования.*

Суть его заключается в том, что каждая прикладная подзадача программируется в отдельном модуле, который отдельно компилируется, тестируется и отлаживается независимо от других модулей. После этого прикладные модули объединяются управляющими модулями, и затем происходит компиляция и отладка всей многомодульной системы. Недостаток этого метода заключается в сложности *организации связей межу модулями* и в *проблемах с исправлением ошибок*, допущенных на ранней стадии программирования. Однако, этот метод приемлем при разработке широкого круга относительно несложных задач.

2. *Метод нисходящего проектирования.*

Этот метод используется при разработке сложных многоуровневых программ. Суть его заключается в том, что программирование начинается с разработки основного управляющего модуля. Затем программируются и подключаются вспомогательные управляющие модули и отлаживаются связи между ними. В конце к разработанной программе подключаются прикладные модули-программы. На каждом из этих этапов происходит общая компиляция и отладка всего комплекса.

Тестирование и отладка программ

Тестирование и отладка написанной программы являются содержанием четвёртого этапа разработки ПО.

Тестирование – выполнение программы с целью обнаружения наличия ошибок.

Тест – совокупность специально подобранных исходных данных и соответствующих им результатов расчетов (как промежуточных, так и окончательных).

Отладка – выполнение программы с целью локализации, диагностики и исправления ошибок.

Причины возникновения ошибок:

- некорректность текста (синтаксические ошибки);
- некорректность компоновки (ошибки редактирования);
- некорректность данных (семантические ошибки);
- некорректность алгоритма (семантические ошибки).

Синтаксические ошибки проявляются на этапе компиляции (система программирования выводит сообщение об ошибке и указывает место в программе, содержащее ошибку).

После компиляции следует компоновка программы, при которой могут быть *ошибки редактирования* (неправильное использование подключаемых модулей).

Семантические ошибки могут проявляться как на этапе выполнения программы (до её завершения), так и после выполнения программы. К первым относятся такие ошибки, как, например, деление на ноль, выход за границы диапазона, нехватка памяти и т.п. О них выводится сообщение компилятором, что облегчает исправление. Семантические ошибки второго типа находить и исправлять гораздо сложнее, так как компилятор их не может найти (они связаны с погрешностями самого алгоритма).

Для поиска этих ошибок используются различные специальные приёмы. Они основаны на получении дополнительной информации о ходе вычислительного процесса.

Некоторые из этих приёмов:

1) *Слежение:*
   - *трассировка* – построчное выполнение программы (клавиши F7, F8 в Turbo Delphi);

- *математическое слежение* – контроль за изменением значений определенных переменных в процессе расчёта (подсказки при наведении курсора на идентификатор при трассировке).

2) *Печать в узлах* – вывод значений заданных переменных в узловых точках программы (разветвление или схождение алгоритма, точки входа и выхода из подпрограммы и др.).

3) *Прокрутка* – вывод значений всех переменных используемых в программе после выполнения каждого оператора в программе.