Лекция Общие сведения о структурной алгоритмизации и технологиях программирования

Цель лекции:

Рассмотреть систематизированные основы знаний по технологиям программирования и структуре языка программирования C++

Учебные вопросы:

1. **Общие сведения о структурной алгоритмизации**
2. **Технологии реализации методов современного программирования**

Список рекомендуемых источников

1 Основы программирования на языках Си и C++ для начинающих [Электронный ресурс].— Режим доступа.— http://cppstudio.com/ .— Загл. с экрана .— Дата последнего обращения 01.10.2016.

2. Портал о программировании. C++ с нуля [Электронный ресурс].— Режим доступа.— https://code-live.ru/tag/cpp-manual/ .— Загл. с экрана .— Дата последнего обращения 01.10.2016.

Этапы решения задач на ЭВМ

Постановка задачи

Формальное построение модели задачи

Построение математической модели задачи

Разработка алгоритма

Программирование

Анализ результатов решения задачи

Сопровождение программы

1 Общие сведения о структурной алгоритмизации

Структурная алгоритмизация воплощает принципы системного подхода в процессе создания и эксплуатации программного обеспечения ЭВМ. В основу структурного программирования положены следующие достаточно простые положения:

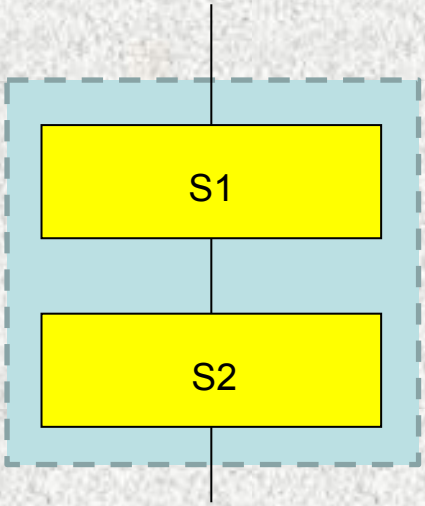
- алгоритм и программа должны составляться поэтапно (по шагам).
- сложная задача должна разбиваться на достаточно простые части, каждая из которых имеет один вход и один выход.
- логика алгоритма и программы должна опираться на минимальное число достаточно простых базовых управляющих структур.

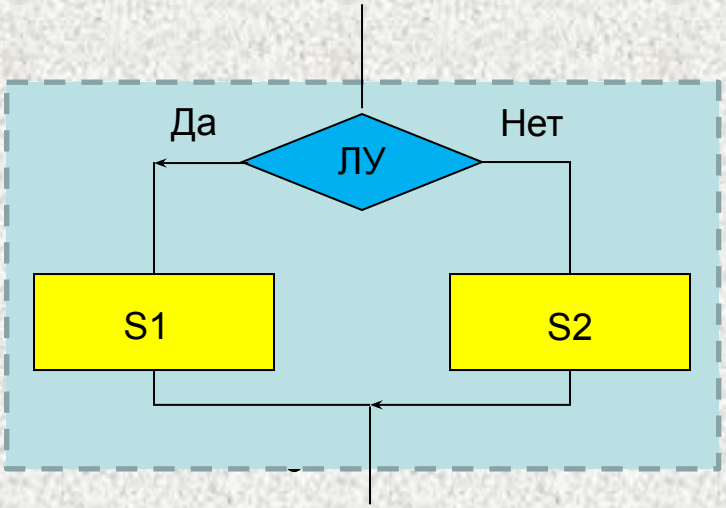
Фундаментом структурного программирования является *теорема о структурировании*.

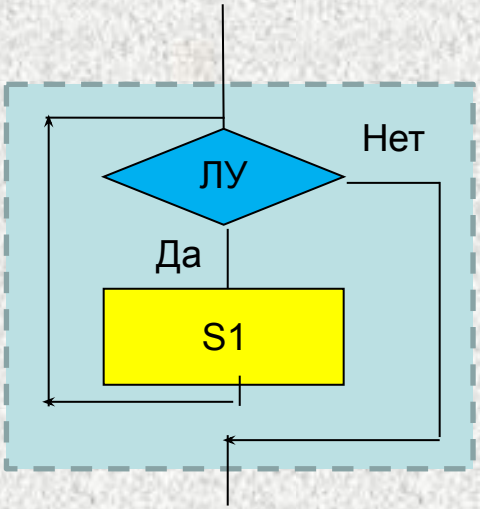
Эта теорема устанавливает, что, как бы сложна ни была задача, алгоритм соответствующей программы всегда может быть представлена с использованием ограниченного числа элементарных управляющих структур.

Базовыми элементарными структурами являются структуры: следование, ветвление и повторение (цикл), любой алгоритм может быть реализован в виде композиции этих трех конструкций.

Основные (базовые) структуры алгоритмов





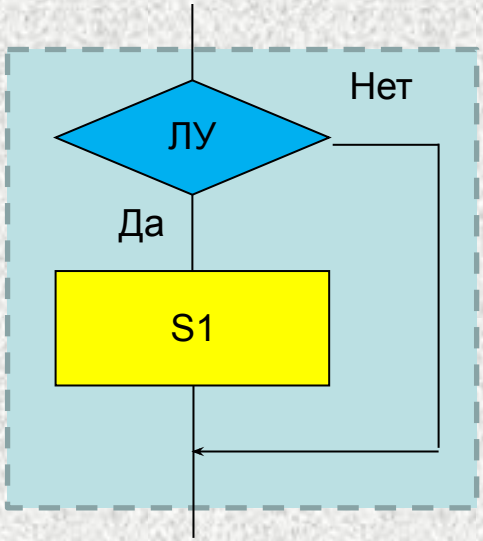


Следование - последовательное выполнение действий (блоков).

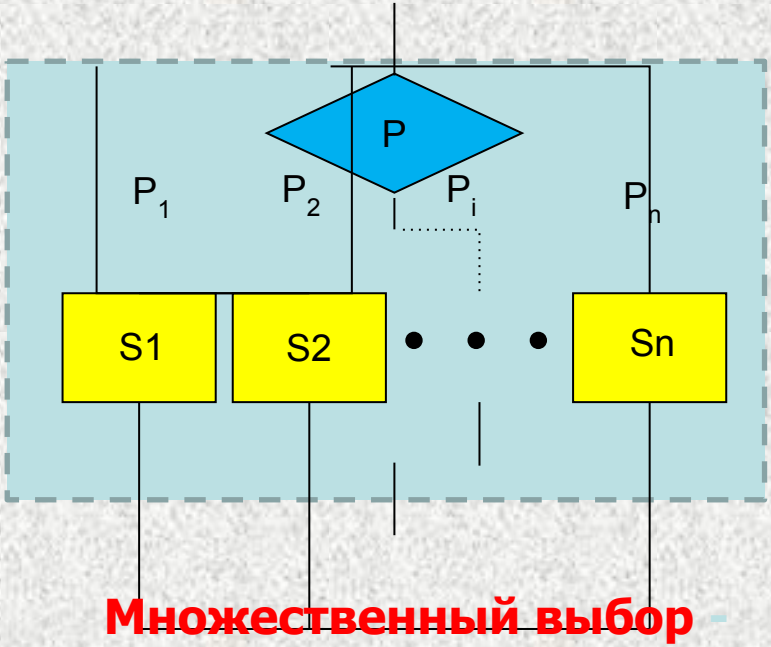
Ветвление - применяется, когда в зависимости от логического условия требуется выполнить одно из двух заранее определенных действий

Цикл с предусловием - пока выполняется логическое условие, осуществляется повторение тела цикла S1.

Дополнительные структуры алгоритмов ветвящихся вычислительных процессов

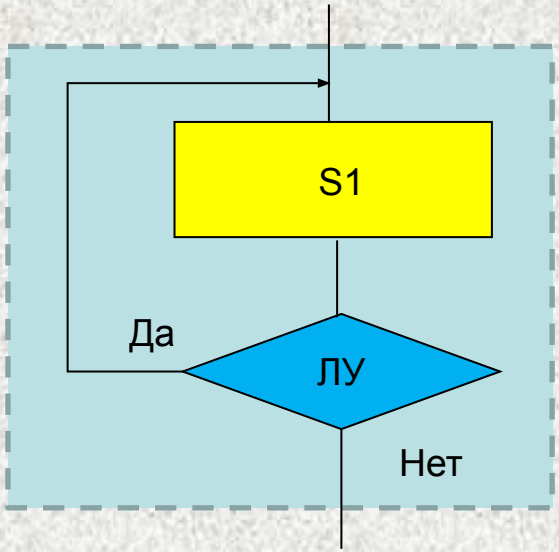


Обход -
частный случай ветвления, когда одна ветвь не содержит ни каких действий

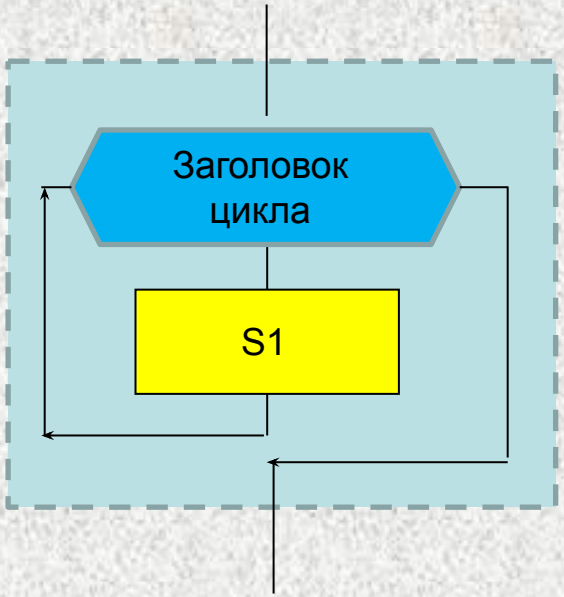


Множественный выбор
обобщение разветвления, когда в зависимости от значения селектора (переключателя Р) выполняется одно из нескольких действий

Дополнительные структуры алгоритмов циклических вычислительных процессов



Цикл с постусловием – пока выполняется логическое условие, осуществляется повторение тела цикла S1.



Цикл с параметром –(цикл с известным числом повторений) заголовок определяет начальное, конечное значение параметра и шаг его изменения

2 Анализ методов решения задач программирования

1 Постановка задачи

На этом этапе формулируется цель решения задачи, анализируются требования и подробно описывается содержание задачи, выявляются условия, при которых решается задача, а также определяются входные параметры, которые называются исходными данными

2 Формальное построение модели задачи

На этом этапе составляется формальная модель решения задачи, например, модель базы данных, адекватная оригиналу, модель объектов и потоков информации

3 Построение математической модели решения задачи

На этом этапе составляется формальная модель решения задачи, например, модель базы данных, адекватная оригиналу, модель объектов и потоков информации

Пример реализации технологии программирования вычислительных задач

Постановка задачи

Разработать программу вычисления суммы квадратов целых положительных чисел от 1 до N

Формальное построение модели задачи

Вычислить:

$S=1+4+9+16+\ldots*N^2$

Построение математической модели решения задачи

1. Исходные данные: **N**
2. Математические формулы:

$$s = \sum_{i=1}^{N} i^2.$$

3. Результат ***S***

Пример реализации технологии программирования вычислительных задач

Постановка задачи

Разработать программу вычисления суммы квадратов целых положительных чисел от 1 до N

Формальное построение модели задачи

Вычислить:

$S=1+4+9+16+\ldots*N^2$

Построение математической модели решения задачи

1. Исходные данные: **N**
2. Математические формулы:

$$s = \sum_{i=1}^{N} i^2.$$

3. Результат ***S***

1. *Ввод с клавиатуры **N** целого типа*
2. *Вычисление **S=0, i=1***

повторять $s=s+i^2$, $i=i+1$ *пока* $i<=N$

3. Печать "S=", S

Пример реализации технологии программирования вычислительных задач

Постановка задачи

Разработать программу вычисления суммы квадратов целых положительных чисел от 1 до N

Формальное построение модели задачи

Вычислить:

$S=1+4+9+16+\ldots*N^2$

Построение математической модели решения задачи

1. Исходные данные: **N**
2. Математические формулы:

$$s = \sum_{i=1}^{N} i^2.$$

3. Результат ***S***

1

1. *Ввод с клавиатуры **N** целого типа*

2

2. *Вычисление **S=0, i=1***

3 4

*повторять **s=s+i²**, **i=i+1** пока **i<=N***

5

Представление математической модели решения задачи в виде вычислительных формул:

1

1. *Ввод с клавиатуры* ***N*** *целого типа*

2

2. *Вычисление* ***S=0, i=1***

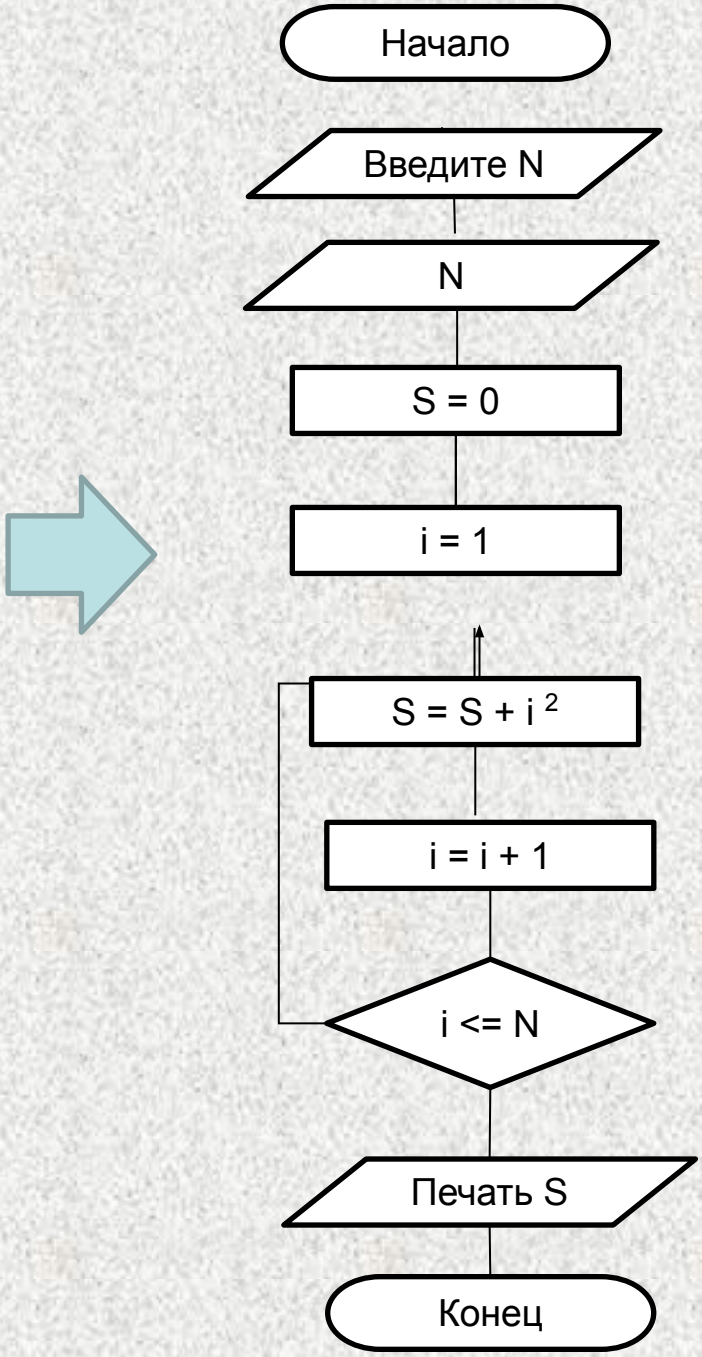
3 4

повторять ***s=s+i², i=i+1*** *пока* ***i<=N***

5

3. *Печать “S=“,* ***S***

позволяет записать алгоритм, представив последовательность выполнения вычислений в виде последовательности базовых структур



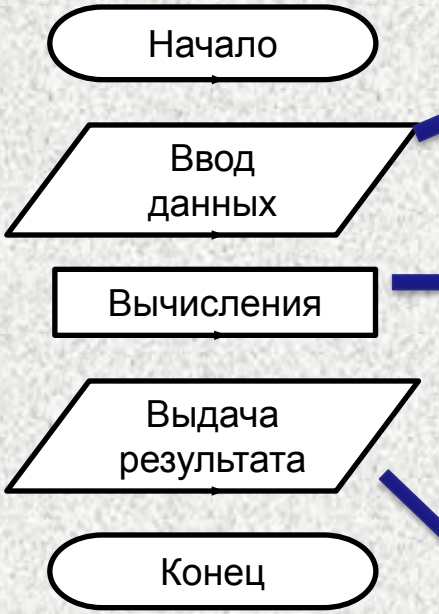
3 Технологии реализации методов разработки алгоритмов и программ

Нисходящее проектирование – технология разработки программ, при которой на каждом шаге проектирования задача разбивается на более мелкие подзадачи так, что в любой момент разработки имеется действующий вариант программы в терминах выделенных подзадач.

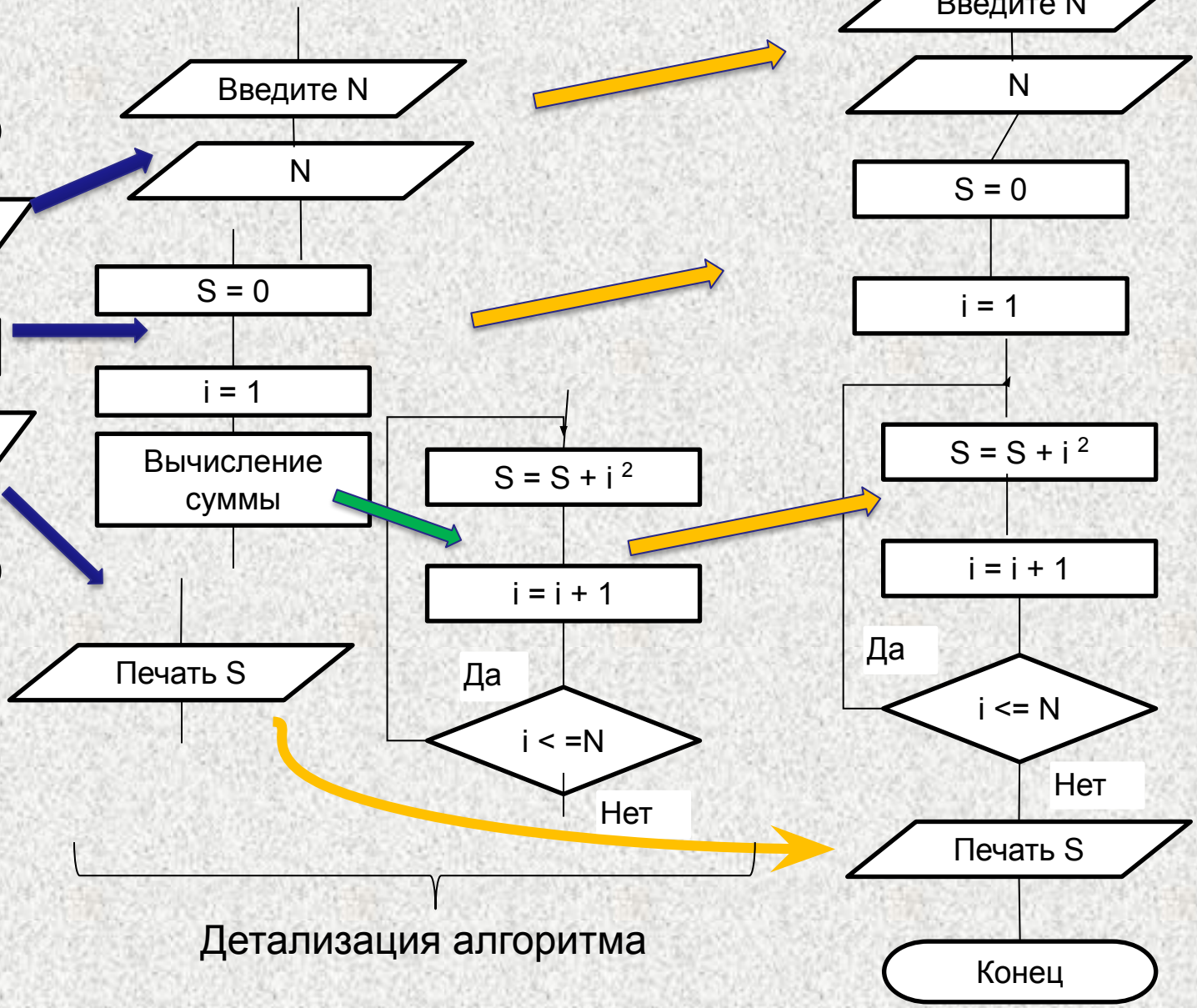
Восходящее проектирование – технология разработки программ, при которой сначала проектируются и отлаживаются подпрограммы для выполнения простых операций, после чего они связываются в единую программу.

Нисходящее проектирование

$$s = \sum_{i=1}^{N} i^2.$$



Общая схема алгоритма



Детализация алгоритма

Основные достоинства нисходящего проектирования:

- проявление логики программы возникает уже при чтении головного модуля, что делает программу боле простой;
- возможность контроля хода работы над программой в процессе последовательной детализации программы обеспечивает ее непрерывную корректировку; отсутствие комплексной отладки благодаря сквозному контролю позволяет сэкономить до 30 % общего времени разработки программ;
- одновременная параллельная работа нескольких программистов может оказаться эффективной.

При нисходящем проектировании, однако, возможны и такие ситуации, когда после значительных затрат на программирование выясняется необходимость объединения нескольких подзадач в один модуль, либо обнаруживается невозможность выполнения модулями нижних уровней своих функций при заданных временных ограничениях.

Восходящее проектирование

Восходящее проектирование (или проектирование «снизу вверх») основано на выделении нескольких достаточно крупных модулей, реализующих некоторые функции в общей программе.

При выделении модулей опираются на доступность реализуемых функций для понимания, простоту структурирования данных, существование готовых программ и модулей для реализации заданных функций, возможности переделки существующих программ для новых целей; имеет значение и размер будущего модуля.

Каждый модуль при восходящем проектировании автономно программируется, тестируется и отлаживается.

После этого отдельные модули объединяются в подсистемы с помощью управляющего модуля, в котором определяется последовательность вызовов модулей, ввод-вывод и контроль данных и результатов.

В свою очередь, подсистемы затем объединяются в более сложные системы и в общий программный комплекс, который подвергается комплексной отладке с проверкой правильности межмодульных связей.