Алгоритмизация и программирование

Парадигмы и стиль программирования

1. Структурное программирование
2. Модульное программирование
3. Объектно-ориентированное программирование
4. Стиль программирования

1

Структурное программирование

Структурное проектирование программных систем

Критерии формирования структуры ПС

Модифицируемость

Отлаживаемость

Удобство управления разработкой

Обеспечение возможности контролируемого изменения конфигурации, состава и функций компонентов с сохранением целостности структурного построения базовых версий ПС.

Эффективное использование памяти или производительности реализующей ЭВМ

Трудоемкость или длительность разработки

Надежность, безопасность и защищенность

Структурное программирование
Основные принципы

Нисходящее функциональное проектирование с выделением в системе основных функциональных подсистем, которые разбиваются на более мелкие подсистемы.

Применение специальных языков проектирования и средств автоматизации использования этих языков.

Планирование и документирование проекта, поддержка соответствия кода проектной документации.

Структурное кодирование без goto.

**Технология структурного программирования
является наиболее правильной с точки зрения управляемости,
безошибочности и качества
получаемого программного кода.**

Структурное программирование
Как правильно использовать goto

```
retry: for(...)   { for (...)
                          {...
                          if () goto retry;... // Повторить все сначала
                          if () goto fatal; }  // Перейти к концу программы
                          }
fatal:
```

При использовании оператора перехода нужно изменить условия текущего выполнения программы применительно к точке перехода (открыть заново файлы, присвоить начальное значение переменным).

Структурное программирование
Операторы continue, break и return

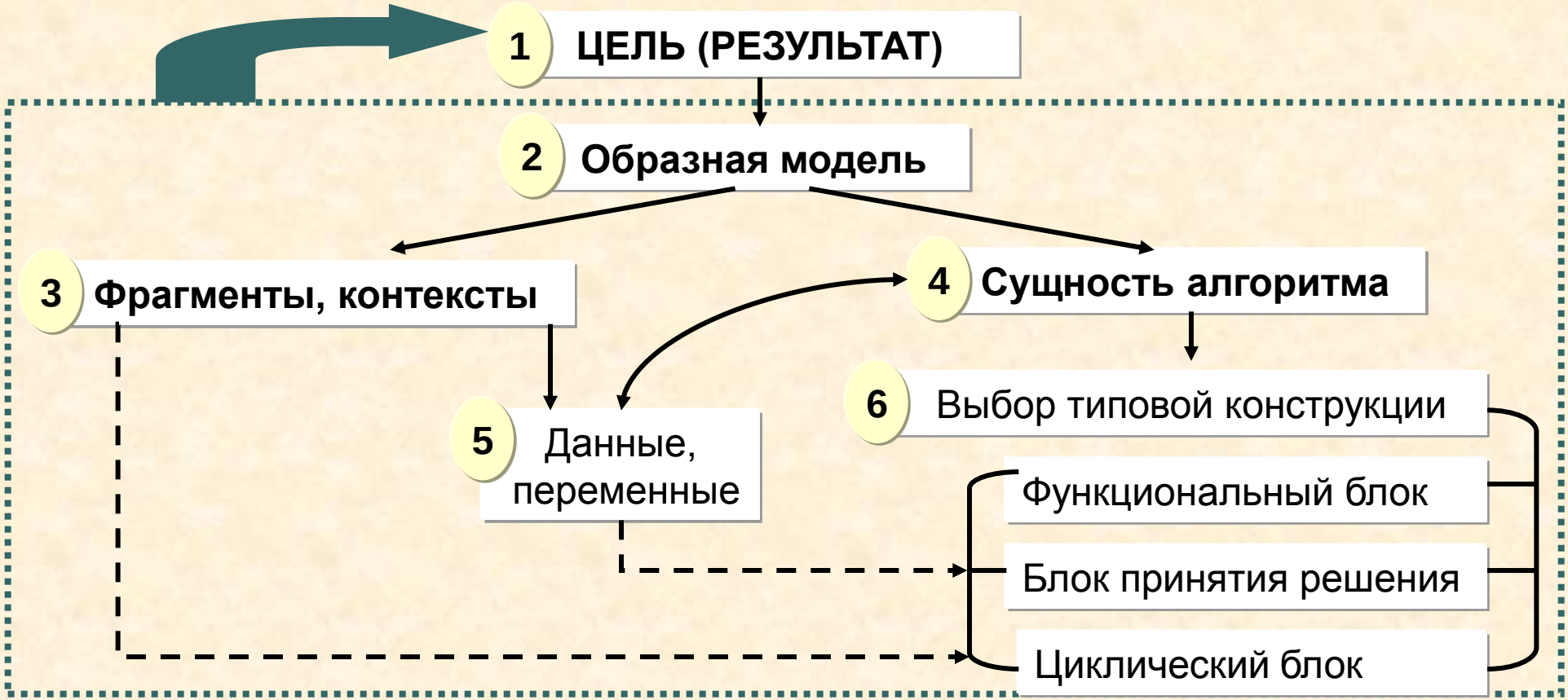
```
void      Fun(){
for (i = 0; i < n; poz1: i++)
          {
          if (A[i] == 0)  continue;      //goto poz1;
          if (A[i] == -1) return;        //goto poz2;
          if (A[i] < 0)  break;          //goto poz3;
          }
poz2:     <продолжение тела функции>
poz3:  }
```

Continue — переход в завершающую часть цикла.
Break — выход из внутреннего цикла.
Return — выход из текущего модуля (функции).

Такие конструкции нарушают чистоту структурного подхода. Однако, все они имеют простые структурированные эквиваленты с использованием дополнительных переменных — признаков.

Структурное проектирование

Этапы



Структурное проектирование
Особенности

На любом промежуточном шаге программа состоит из конструкций языка программирования и словесных формулировок.

Конструкции языка соответствуют пройденным шагам проектирования.

Словесные формулировки соответствуют еще не реализованным вложенным конструкциям нижнего уровня.

Процесс проектирования заключается в последовательной замене словесных формулировок конструкциями языка.

На каждом шаге в программу добавляется всего одна конструкция.

Содержимое составных частей конструкции формулируется в терминах **цель** или **результат**.

Свобода выбора ограничена тремя управляющими конструкциями: последовательность действий, ветвление, цикл.

2

Модульное программирование

Иерархические уровни в структуре ПС

ПРОГРАММНЫЕ МОДУЛИ
(оформляются как законченные компоненты текста программ)

ФУНКЦИОНАЛЬНЫЕ ГРУППЫ
(компоненты или пакеты программ)

КОМПЛЕКСЫ ПРОГРАММ
(оформляются как законченные программные продукты определенного целевого назначения)

Что такое модульность

Модульность является важным качеством инженерных процессов и продуктов.

Модульность позволяет применять принцип разделения задач при работе
с элементами каждого модуля отдельно,
с общими характеристиками групп модулей,
с отношениями между группами модулей.

Проектирование **снизу вверх**. Процессы сначала концентрируются на модулях, а затем на их объединении.

Проектирование **сверху вниз**. Сначала систему разбивают на модули, а потом работают над их индивидуальным проектированием.

Проектирование программных модулей

Разработка

Локальные функции и алгоритмы обработки данных

Межмодульные интерфейсы

Внутренние структур данных

Структурные схемы передач управления

Средства управления в исключительных ситуациях

Функции

Порядок следования отдельных шагов обработки

Ситуации и типы данных, вызывающие изменения процесса обработки

Повторно используемые функции программы

Для многократного использования программных модулей нужно применять унифицированные правила структурного построения, оформления спецификаций требований, описаний текстов программ и комментариев.

Модульное программирование

Главный принцип модульного программирования — выделение ранее написанных фрагментов и оформлении их в виде **модулей**.

Каждый модуль снабжается описанием, в котором устанавливаются правила его использования — интерфейс модуля.

Интерфейс задает связи модуля с основной программой — связи по данным и связи по управлению.

Пример. Модули решения математических задач (уравнения, системы уравнений, задачи оптимизации и пр.).

Если некоторая задача в процессе проектирования порождает значительное число программных контекстов и имеет высокий уровень вложенности управляющих конструкций, то в ней можно выделить подзадачу, оформив ее в виде функции.

Модульное программирование
Принципы

Логическая завершенность. Функция (модуль) должна реализовывать логически законченный, целостный алгоритм.

Ограниченность. Функция (модуль) должна быть ограничена в размерах.

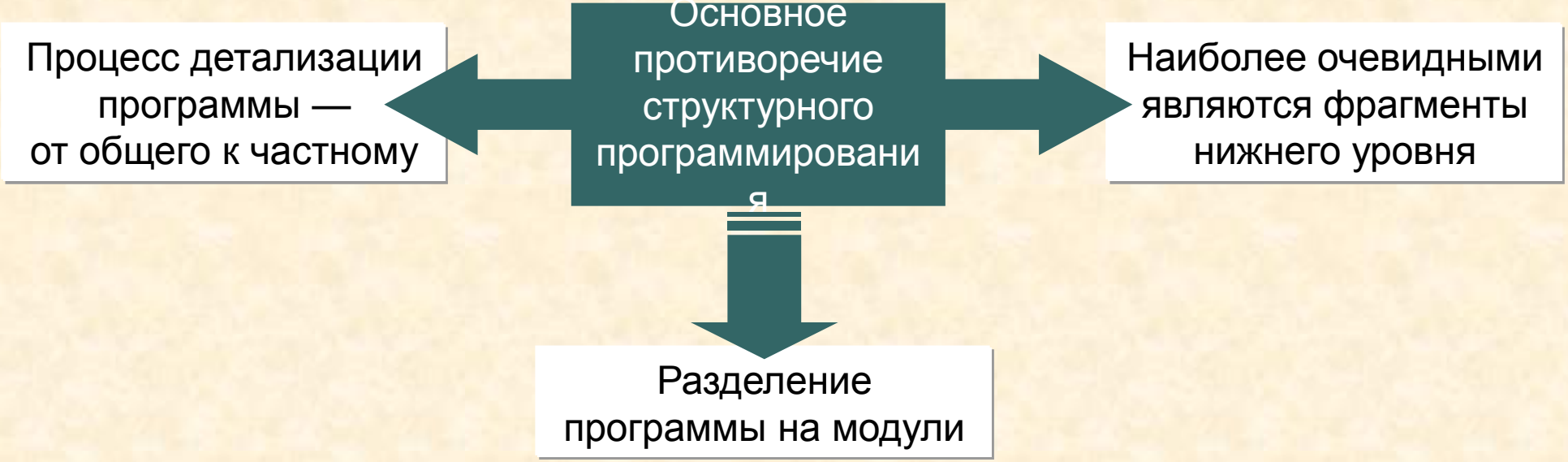
Замкнутость. Функция (модуль) не должна использовать глобальные данные, не должна содержать ввода и вывода результатов во внешние потоки. Результаты должны быть размещены в структурах данных.

Универсальность. Функция (модуль) должна быть универсальна.

Принцип «черного ящика». Функция (модуль) должна иметь продуманный программный интерфейс.

Программный интерфейс *— набор фактических параметров и результат функции, через который она вызывается в других частях программы.*

Модульное структурное восходящее программирование



При проектировании программного комплекса «снизу-вверх» можно послойно наращивать набор функций.

Каждая вышележащая функция использует разработанные ранее функции.

Принципы структурного проектирования каждой отдельной функции сохраняются.

3

Объектно-ориентированное программирование

Объектно-ориентированное программирование

Вводится понятие класса как развитие понятия модуля с определенными свойствами и поведением, характеризующими обязанностями **класса**.

Каждый класс может порождать **объекты** — экземпляры данного класса.

Основные принципы ООП

Инкапсуляция — объединение в классе данных (свойств) и методов (процедур обработки).

Наследование — возможность вывода нового класса из старого с частичным изменением свойств и методов.

Полиморфизм — определение свойств и методов объекта по контексту.

4

Стиль программирования

Стиль программирования
Синдром «копировать/вставить»

**Что хуже, отладить один и тот же код дважды
или найти ошибку только в одном куске скопированного кода?**

Ввод/вывод, как правило, буферизуется операционной системой.

Использование **printf** в процессе отладки рискованно.

Используйте отладчик для выяснения, в каких строках кода ошибка.

Всегда проверяйте работу вспомогательных функций. По возможности, изолируйте вспомогательные функции и проверьте каждую отдельно.

Стиль программирования
Читабельность

Ясность кода. Все, что вы пишите, вам придется перечитывать.

Пробелы улучшают читабельность, используются для форматирования отступов, пространства вокруг операторов, подписи функций и размещения аргументов функций.

Отступы помогают быстро находить нужные управляющие строки кода . Следует делать отступы в каждом блоке кода.

Ширина отступа (2 – 8 пробелов). Лучшее решение для слишком длинных строк — снизить сложность кода или выделить в отдельные некоторые функции.

Скобки должны позволять разместить максимальное количество кода на экране при соблюдении последовательности.

Стиль программирования
Разделение сложных выражений

Легче отслеживать выражение.

Можно дать уникальное имя каждому промежуточному шагу, что повысит читабельность кода.

Можно повторно использовать промежуточные вычисления.

Можно написать больше комментариев в строке для объяснения что происходит и почему.

**Чем меньше событий на одной строке кода,
тем легче видеть то, что происходит.**

Стиль программирования
Числа в коде

Лучшее решение — использование макросов в C или констант в C++.

Возможность наглядно называть числа.

Облегчает обнаружение использования определенного числа и различать числа с тем же значением, но разным смыслом.

При изменении значения есть всего одно место, куда нужно внести изменения, вместо того, чтобы просматривать все точки вхождения.