КЛАССИЧЕСКИЕ МЕТОДЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ВОПРОСЫ:

1. Особенности процесса проектирования программного обеспечения
2. Классические методы проектирования модульных программных средств
3. Оценка структурного разбиения программы на модули

ОСОБЕННОСТИ ПРОЦЕССА ПРОЕКТИРОВАНИЯ ПО

- Проектирование объясняет, как ПС будет удовлетворять предъявленным к нему требованиям.
- Проектирование — итерационный процесс, при помощи которого требования к ПС транслируются в инженерные представления ПС.

ОСОБЕННОСТИ ПРОЦЕССА ПРОЕКТИРОВАНИЯ ПО

КЛАССИЧЕСКИЕ МЕТОДЫ СТРУКТУРНОГО ПРОЕКТИРОВАНИЯ МОДУЛЬНЫХ ПС

1. Методы нисходящего проектирования
2. Методы расширения ядра
3. Методы восходящего проектирования.

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

- На каждом шаге нисходящего проектирования делается оценка правильности вносимых уточнений в контексте правильности функционирования разрабатываемого программного средства в целом.
- На каждом шаге разработки уточняется реализация фрагмента алгоритма, то есть решается более простая задача.
- Компоненты нижнего уровня ПС называются программными модулями.
- Метод нисходящего проектирования положен в основу стандартного процесса разработки, регламентированного стандартом СТБ ИСО/МЭК 12207–2003.

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

Основные классические стратегии, на которых основана реализация метода нисходящего проектирования:

- пошаговое уточнение (Дейкстра)
- анализ сообщений (Йодан, Константайн, Майерс).

Эти стратегии отличаются способами определения начальных спецификаций требований, методами разбиения задачи на части и правилами записи (нотациями), положенными в основу проектирования ПС.

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

Способы реализации пошагового уточнения:

1) проектирование программного средства с помощью псевдокода и управляющих конструкций структурного программирования;
2) использование комментариев для описания обработки данных.

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

Пример. Пусть программа обрабатывает файл дат. Необходимо отделить правильные даты от неправильных, отсортировать правильные даты, перенести летние и зимние даты в выходной файл, вывести неправильные даты.

1 этап пошагового уточнения. Задается заголовок программы, соответствующий ее назначению.

Program Обработка_дат.

End.

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

2 этап пошагового уточнения. Определяются основные структурные компоненты программы в соответствии с ее основными функциями

```
Program Обработка_дат;
    Отделить_правильные_даты_от_неправильных {*}
    Сортировать_правильные_даты Выделить_зимние_и_летние_даты
    Обработать_неправильные_даты
End.
```

.

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

3 этап пошагового уточнения. Дальнейшая детализация программы. Детализация фрагмента {*}.

```
Program Обработка_дат;
    While не_конец_входного_файла Do
        Begin
            Прочитать_дату
            Проанализировать_правильность_даты
        End
    Сортировать_правильные_даты
    Выделить_зимние_и_летние_даты
    Обработать_неправильные_даты
End.
```

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

2) При этом способе на каждом этапе уточнений используются управляющие конструкции структурного программирования, а правила обработки данных не детализируются. Они описываются в виде комментариев.

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

Наиболее часто используются следующие виды комментариев:

- заголовки. Объясняют назначение основных компонентов программы на отдельных этапах пошаговой детализации;
- построчные. Описывают мелкие фрагменты программы;
- вводные. Помещаются в начале текста программы и задают общую информацию о ней (например, назначение программы, сведения об авторах, дата написания, используемый метод решения, время выполнения, требуемый объем памяти).

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

Пример. Тот же

1 этап пошагового уточнения. Записываются вводный комментарий и комментарий к заголовку программы.

{Программа обработки дат. Разработчик Иванов И. И.}
{Заголовок программы}

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

2 этап пошагового уточнения. Определяются основные шаги обработки дат. Сохраняются комментарии предыдущего этапа и добавляются комментарии текущего этапа.

```
{Программа обработки дат. Разработчик Иванов И. И.}
{Заголовок программы}
 Program {Обработка дат}
    {Отделение правильных дат от неправильных} {*}
    {Сортировка правильных дат}
     {Выделение зимних и летних дат}
    {Обработка неправильных дат}
End.
```

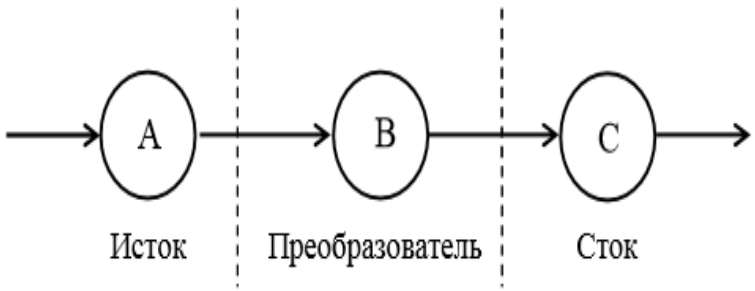
МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

3 этап пошагового уточнения. Дальнейшая детализация программы. Детализация фрагмента {*}.

```
{Программа обработки дат. Разработчик Иванов И. И.}
 {Заголовок программы}
Program {Обработка дат}
     {Отделение правильных дат от неправильных} {*}
         While {не конец входного файла} Do
         Begin
              {Чтение даты}
               {Анализ правильности даты}
         End
    {Сортировка правильных дат}
    {Выделение зимних и летних дат}
    {Обработка неьнправилых дат}
End.
```

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

- Анализ сообщений используется в первую очередь для структуризации ПС обработки информации и основывается на **анализе потоков данных**, обрабатываемых программным средством.
- Первоначальный поток данных разбивается на три потока: первый содержит непреобразованные входные данные, второй – потоки преобразования, третий – только выходную информацию.



МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

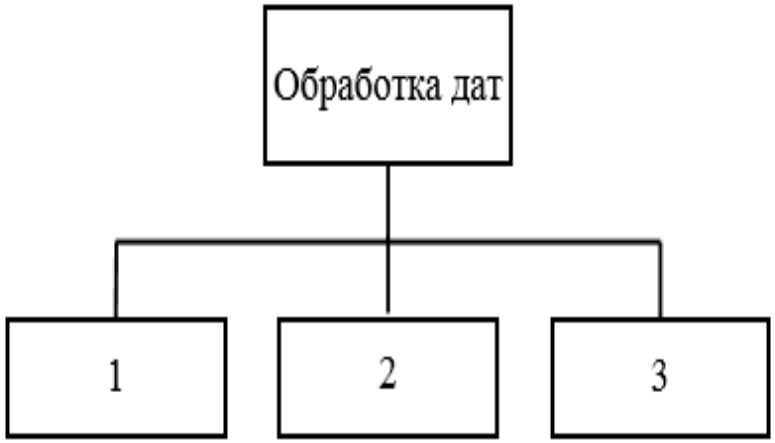
Части программы, соответствующие трем потокам данных:

- **исток** (выполняет функцию управления входным потоком данных)
- **преобразователь** (основная часть программы)
- **сток** (выполняет функцию управления выходным потоком данных).

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

- Процесс декомпозиции программы заключается в рекурсивном использовании метода разбиения на ***исток – преобразователь – сток*** на отдельных ветвях ее древовидной структуры.

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

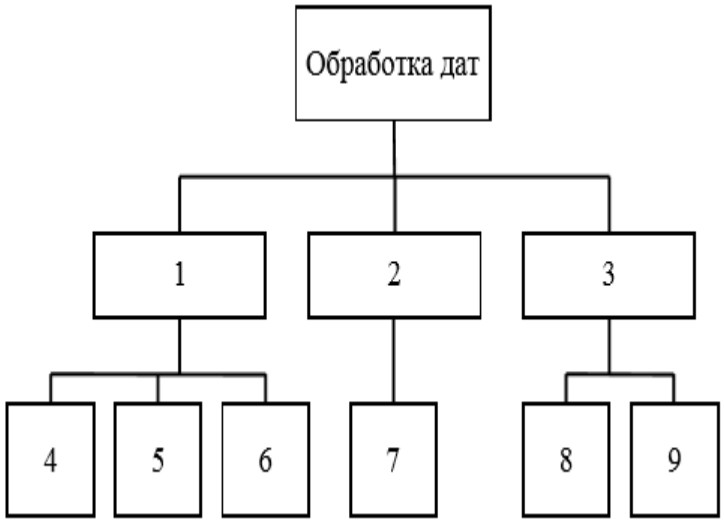


1 – проверка входных дат (исток);

2 – обработка дат (преобразователь);

3 – запоминание правильных результатов (сток);

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ



4 – чтение дат (исток истока 1);
5 – проверка дат (преобразователь истока 1);
6 – сохранение неправильных дат (сток истока 1);
7 – обработка правильных дат (преобразователь преобразователя 2);
8 – проверка результата (преобразователь стока 3);
9 – запоминание дат (сток стока 3).

МЕТОДЫ НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

Таблица связей между компонентами

Компонент	Вход	Выход
1		Правильные даты
2	Правильная дата	Результаты
3	Результаты	
4		Прочитанная дата
5	Прочитанная дата	Неправильная дата Правильная дата
6	Неправильная дата	
7	Правильная дата	Обработанная дата
8	Обработанная дата	Неправильный результат Правильный результат
9	Правильная дата	

МЕТОД ВОСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

Метод восходящего проектирования целесообразно применять в следующих случаях:

- существуют разработанные модули, которые могут быть использованы для выполнения некоторых функций разрабатываемой программы;
- заранее известно, что некоторые простые или стандартные модули потребуются нескольким различным частям программы (например, подпрограмма анализа ошибок, ввода-вывода и т.п.).

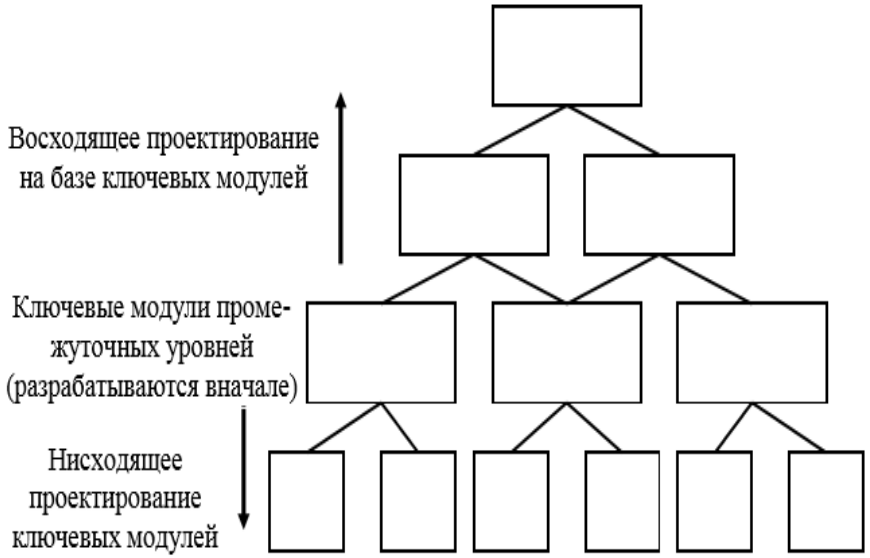
МЕТОД ВОСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

Способы сочетания методов нисходящего и восходящего проектирования:

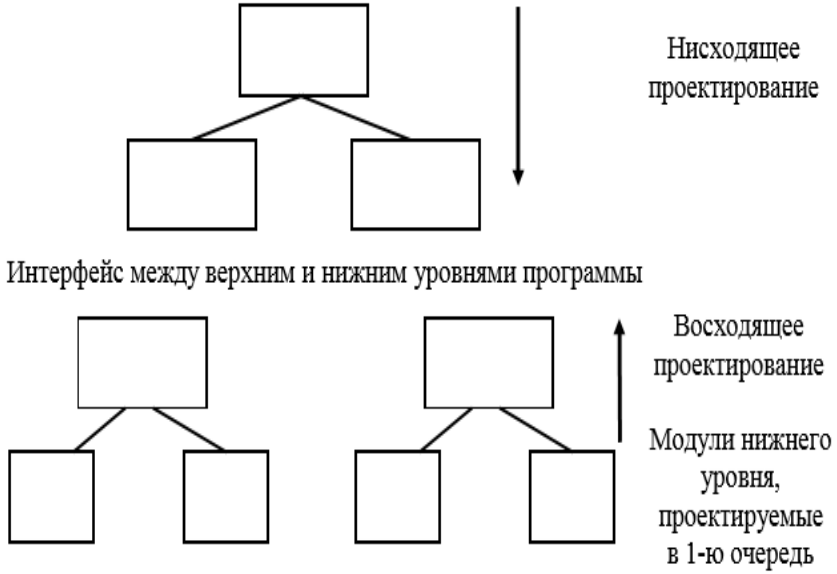
I. Выделяются ключевые (наиболее важные) модули промежуточных уровней разрабатываемой программы. Затем проектирование ведется нисходящим и восходящим методами одновременно

II. Проектируются модули нижнего уровня (те, которые необходимо спроектировать заранее). Затем программа проектируется одновременно нисходящим и восходящим методами.

МЕТОД ВОСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ



Первый способ



Второй способ

МЕТОДЫ РАСШИРЕНИЯ ЯДРА

Подходы к реализации методов расширения ядра:

1) основан на методах проектирования структур данных, используемых при иерархическом проектировании модулей. Данный подход применяется в методах, разработанных Джексоном;
2) основан на определении областей хранения данных с последующим анализом связанных с ними функций

МЕТОД JSP (МЕТОД ДЖЕКСОНА)

- базируется на положении, что структура программы зависит от структуры подлежащих обработке данных

→ структура данных может использоваться для формирования структуры программы.

МЕТОД JSP (МЕТОД ДЖЕКСОНА)

Этап анализа

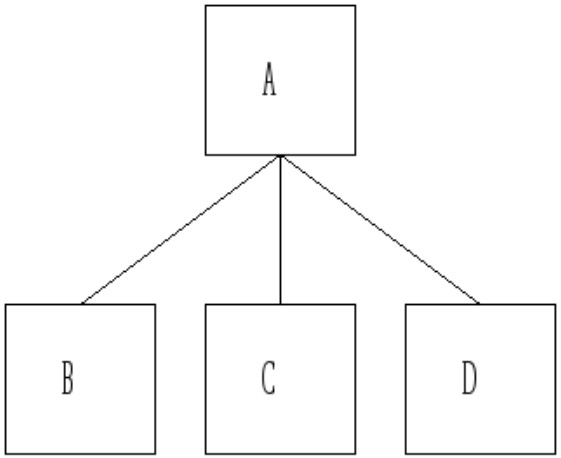
1) *Объект-действие.* Определяются объекты — источники или приемники информации и действия — события реального мира, воздействующие на объекты.
2) *Объект-структура.* Действия над объектами представляются диаграммами Джексона.
3) *Начальное моделирование.* Объекты и действия представляются как обрабатывающая модель. Определяются связи между моделью и реальным миром.

МЕТОД JSP (МЕТОД ДЖЕКСОНА)

4) *Доопределение функций.* Выделяются и описываются сервисные функции.
5) *Учет системного времени.* Определяются и оцениваются характеристики планирования будущих процессов.
6) *Реализация.* Согласование с системной средой, разработка аппаратной платформы.

МЕТОД JSP (МЕТОД ДЖЕКСОНА)

Конструкция последовательности данных: два или более компонента данных следуют друг за другом строго последовательным образом и образуют единый компонент данных.

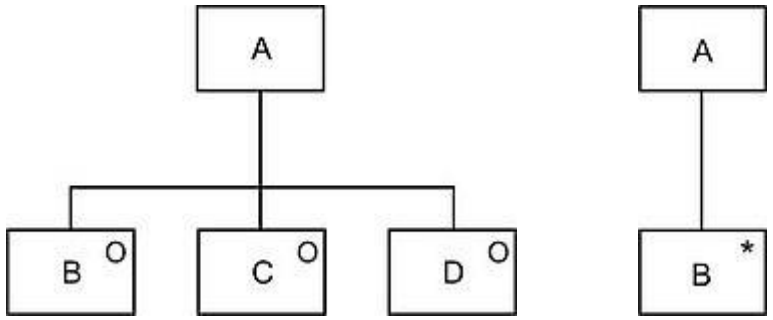


Элементарная конструкция: элементарными являются те компоненты, которые не разбиваются далее на подкомпоненты.

МЕТОД JSP (МЕТОД ДЖЕКСОНА)

Конструкция выбора данных: конструкция сведения результирующего компонента данных к одному из двух или более выбираемых подкомпонентов.

Конструкция повторения данных : конкретный элемент данных может повторяться от нуля до неограниченного числа раз.



ПРИМЕР

Разработать компьютерную систему для обслуживания университетских перевозок. Университет размещается на двух территориях. Для перемещения студентов используется один транспорт. Он перемещается между двумя фиксированными остановками. На каждой остановке имеется кнопка вызова.

При нажатии кнопки:

- если транспорт на остановке, то студенты заходят в него и перемещаются на другую остановку;
- если транспорт в пути, то студенты ждут прибытия на другую остановку, приема студентов и возврата на текущую остановку;
- если транспорт на другой остановке, то он ее покидает, прибывает на текущую остановку и принимает студентов, нажавших кнопку.

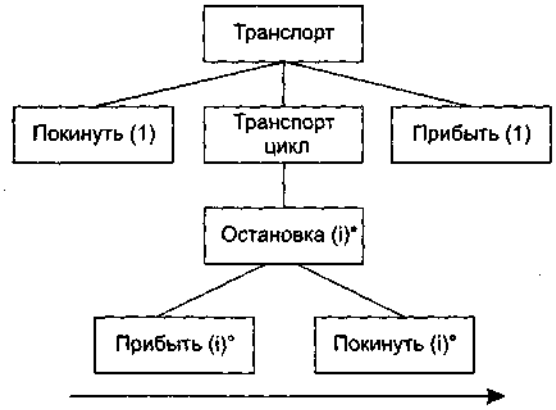
Транспорт должен стоять на остановке до появления запроса на обслуживание.

ПРИМЕР

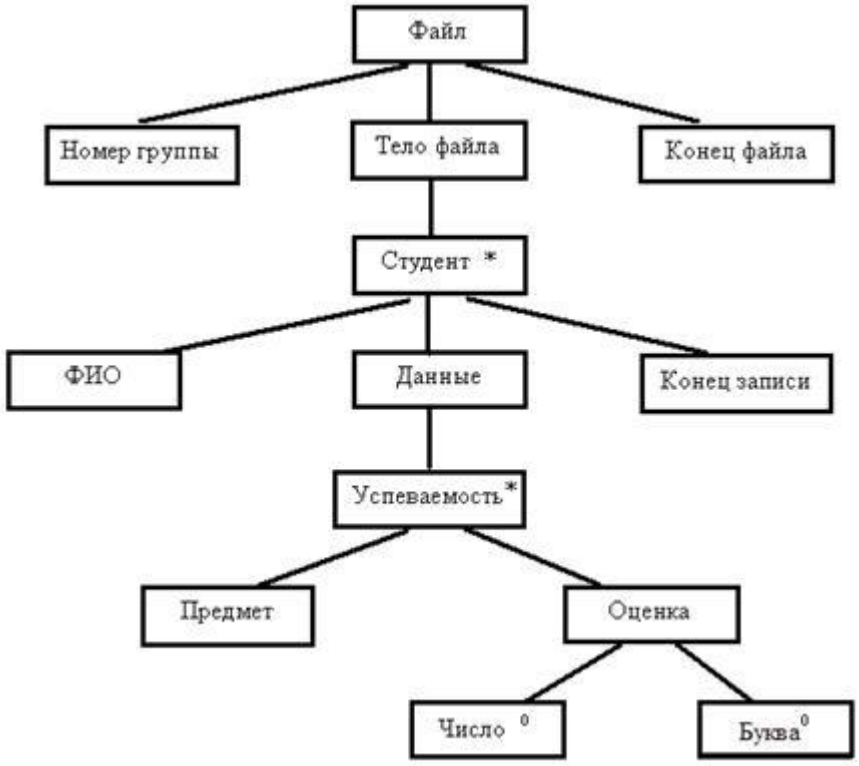
1) *Объект-действие:*

- Транспорт- прибывать, покидать;
- кнопка – нажимать.

2) *Объект-структура*



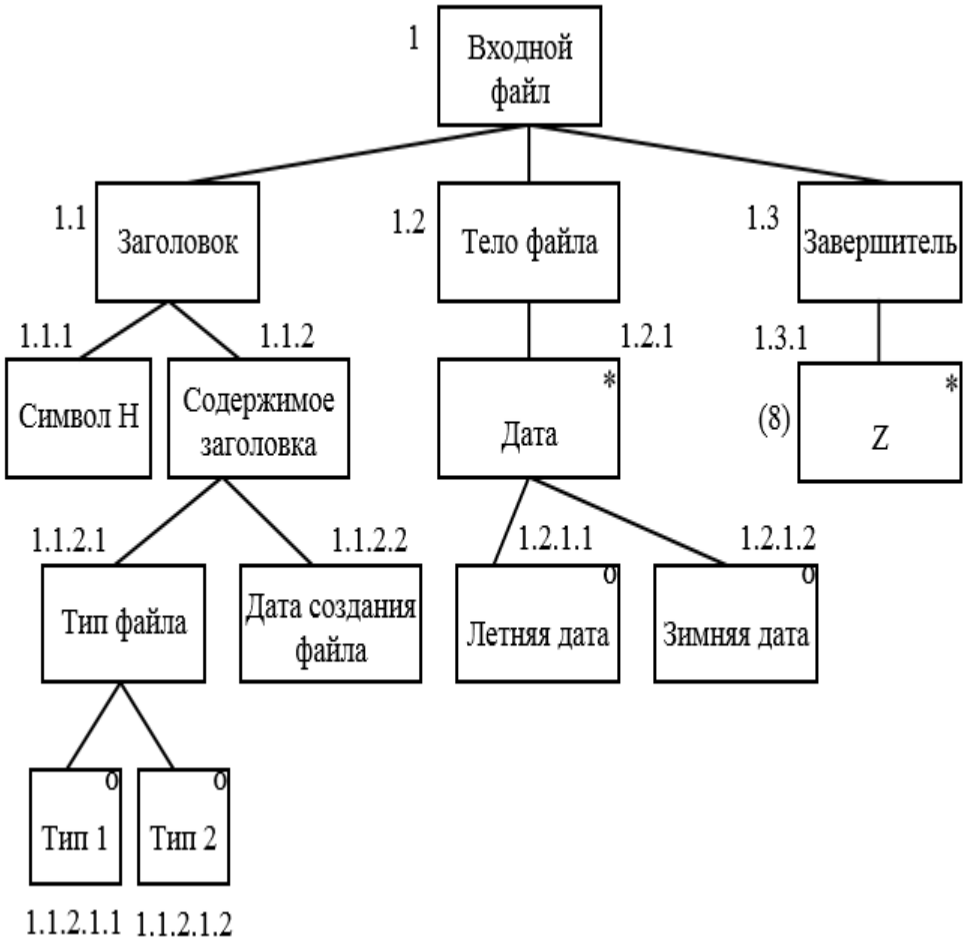
ПРИМЕР

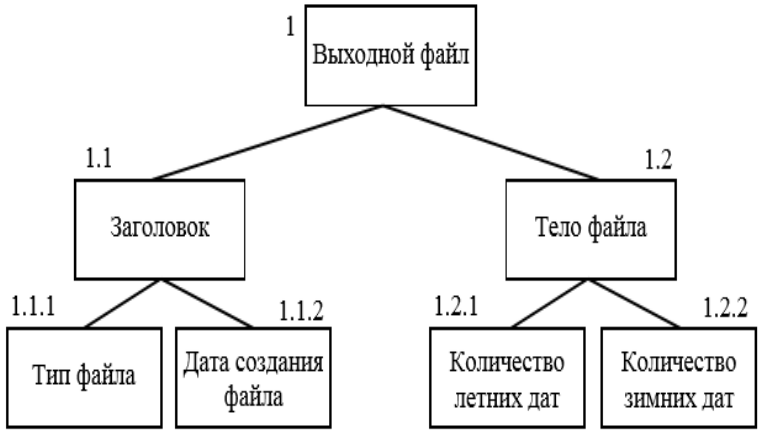

Файл
Номер группы
Тело файла
Конец файла
Студент *
ФИО
Данные
Конец записи
Успеваемость*
Предмет
Оценка
Число 0
Буква0

МЕТОДЫ РАСШИРЕНИЯ ЯДРА

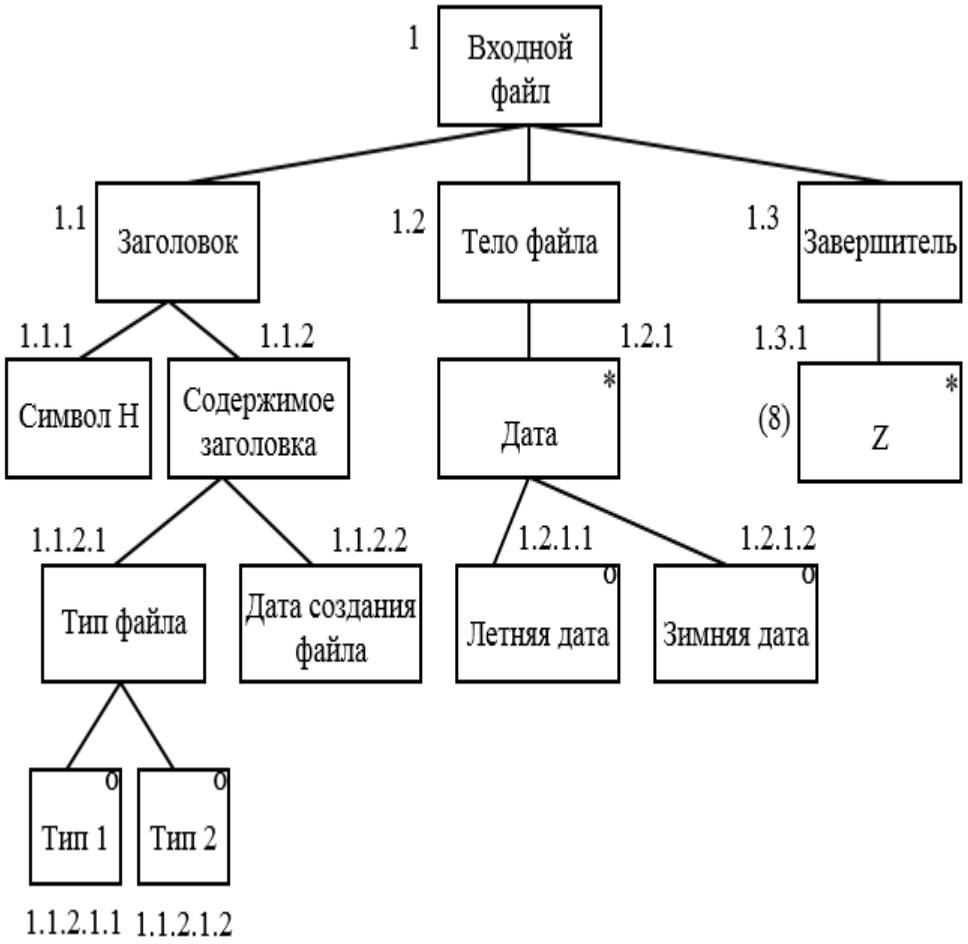
1. Проектирование структур входных и выходных данных (*Объект – действие*).
2. Идентификация соответствий между структурами данных (*Объект – структура*).
3. Проектирование структуры программы.
4. Перечисление и распределение выполняемых операций.
5. Создание текста программы на метаязыке структурированного описания.

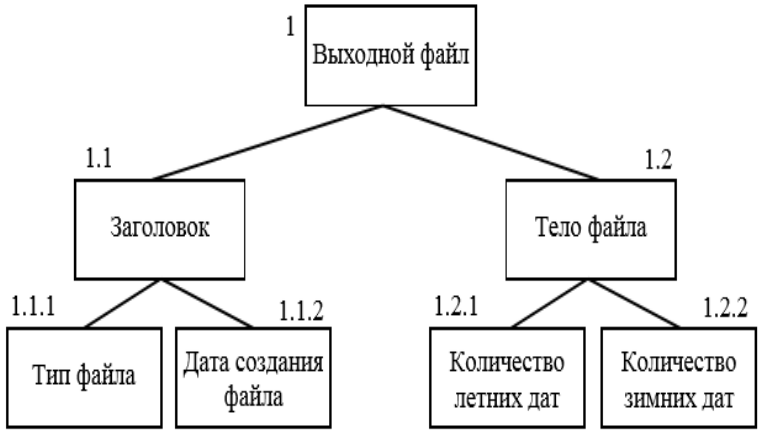
МЕТОДЫ РАСШИРЕНИЯ ЯДРА





МЕТОДЫ РАСШИРЕНИЯ ЯДРА





МЕТОДЫ РАСШИРЕНИЯ ЯДРА