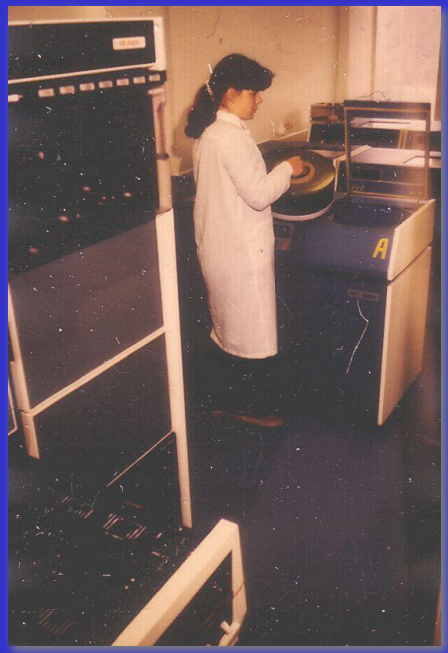
Тема 3. Структурное программирование

На заре программирования...



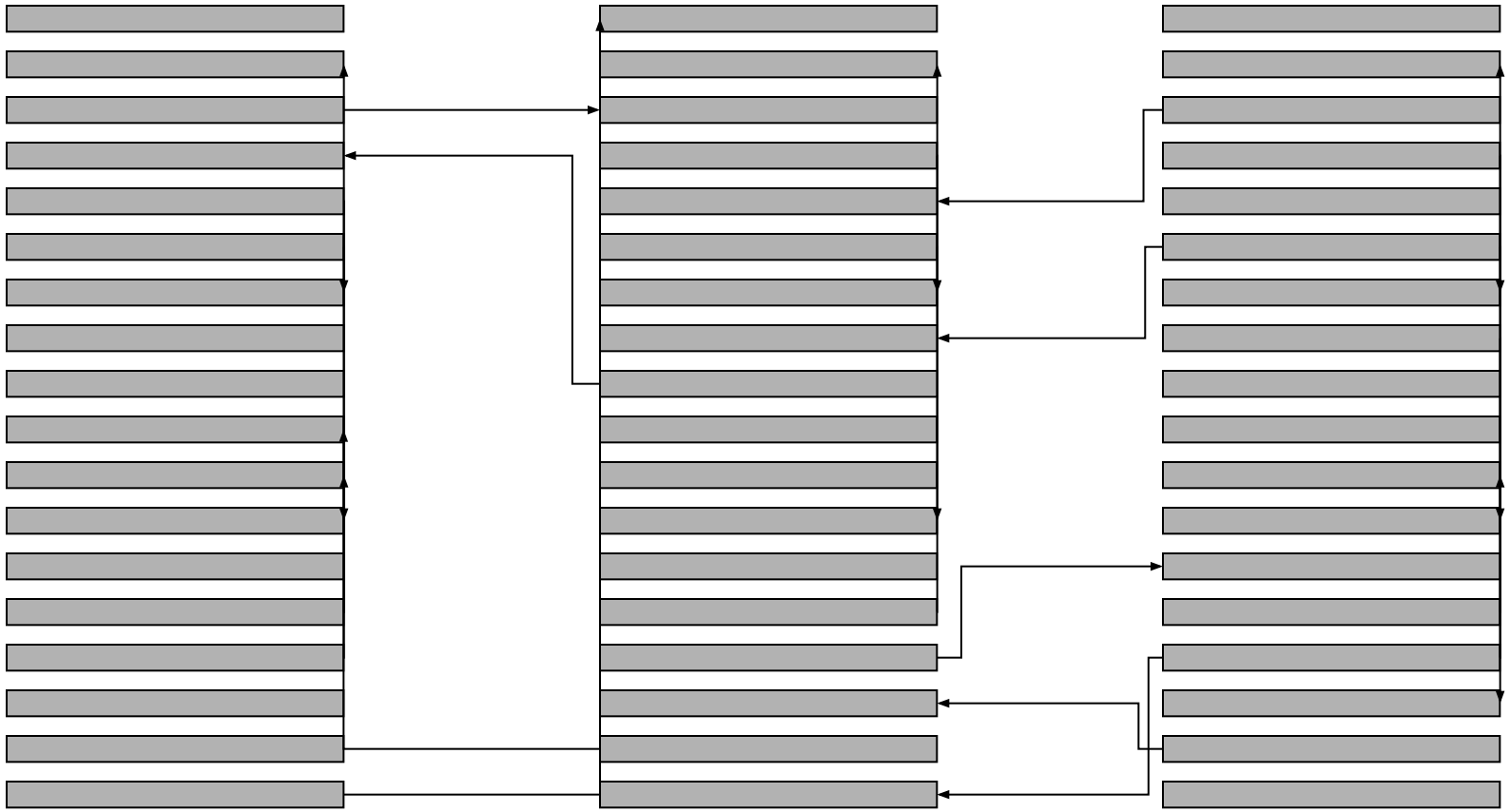
```
QTRAN::    .IRPC     X,<1234>
           MOV       R'X,-(SP)
           .ENDR
           MOV       Q.RTBA(R5),R0
           MOV       Q.RTBS(R5),R1
1$:        CLR       TR.RLC(R0)
           ADD       #TR.SIZ,R0
           SOB       R1,1$
           MOV       Q.ATBA(R5),R0
           MOV       Q.ATBS(R5),R1
           MUL       #TA.SIZ,R1
2$:        CLRB        (R0)+
           SOB       R1,2$
```

Что такое хорошая программа?

Раньше хорошими программистами считали тех, кто писал весьма хитроумные программы, которые занимали минимум оперативной памяти и выполнялись за кратчайшее время. Это было естественно, потому что в "старое доброе время" размер оперативной памяти был сильно ограничен, а машины были намного медленнее, чем сегодня. Результатом хитроумного кодирования оказывались программы, которые было трудно понять другим лицам. Программисты зачастую сами признавали, что свою собственную программу они с трудом понимают уже через полгода, а то и через месяц.

Дж. Хьюз, Дж. Мичтом. Структурный подход к программированию

Проблема "блюда спагетти"



Разработка программ подчиняется законам Мэрфи

Законы Мэрфи

Всё сложнее чем кажется.

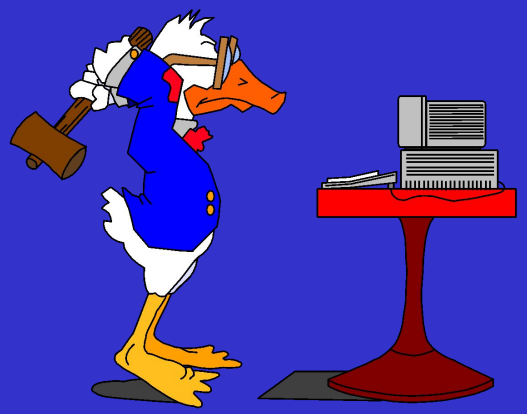
Всё тянется дольше, чем можно ожидать.

Всё оказывается дороже, чем планировалось.

Если что-то может испортиться, оно обязательно портится.

Комментарий Каллагана к законам Мэрфи

Мэрфи был оптимистом.



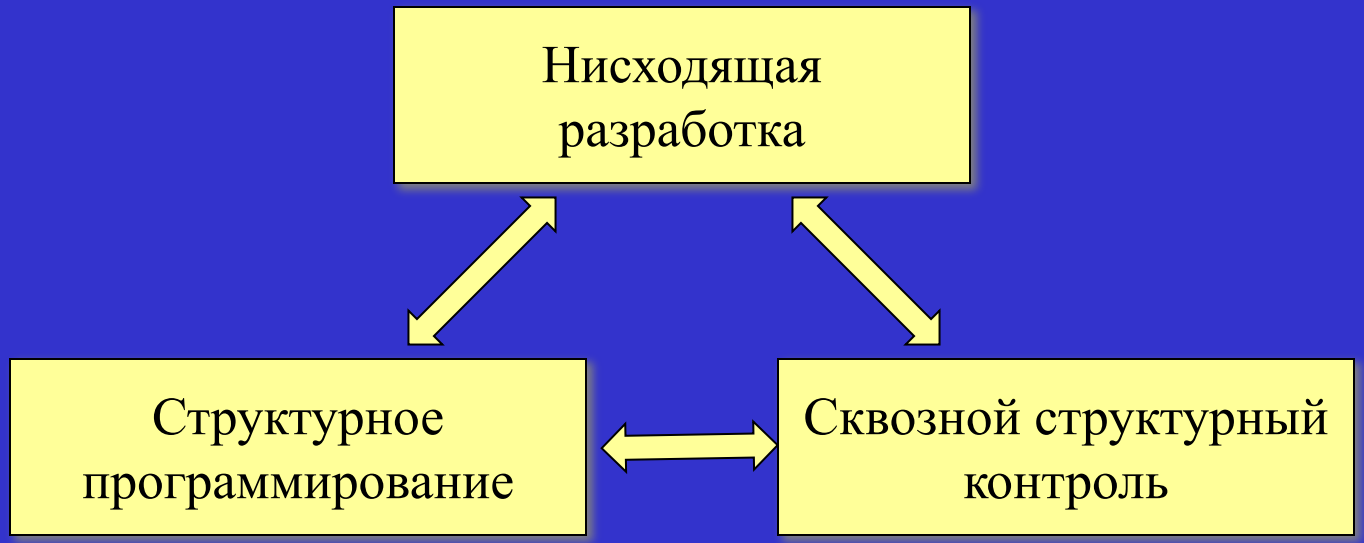
Предпосылки структурного программирования

"На протяжении многих лет я очень хорошо знал, что квалификация программистов - убывающая функция от плотности операторов GOTO в создаваемых ими программах. Но лишь совсем недавно я обнаружил, почему использование оператора GOTO имеет такие гибельные последствия. Я пришел к убеждению, что этот оператор должен быть исключён из всех языков программирования высокого уровня.

Эдсгер Дейкстра, март 1968

Структурное программирование - программирование без GOTO

Структурный подход к программированию



Цель - разработка понятных, правильных, легко сопровождаемых программ

Нисходящая разработка

Разработка «сверху-вниз» по модульному принципу

Свойства модуля

1. Модуль может быть отдельной программой или подпрограммой (функцией).

2. На модуль можно ссылаться с помощью имени, называемого именем модуля.

3. Модуль должен возвращать управление тому, кто его вызвал.

4. Модуль может обращаться к другим модулям.

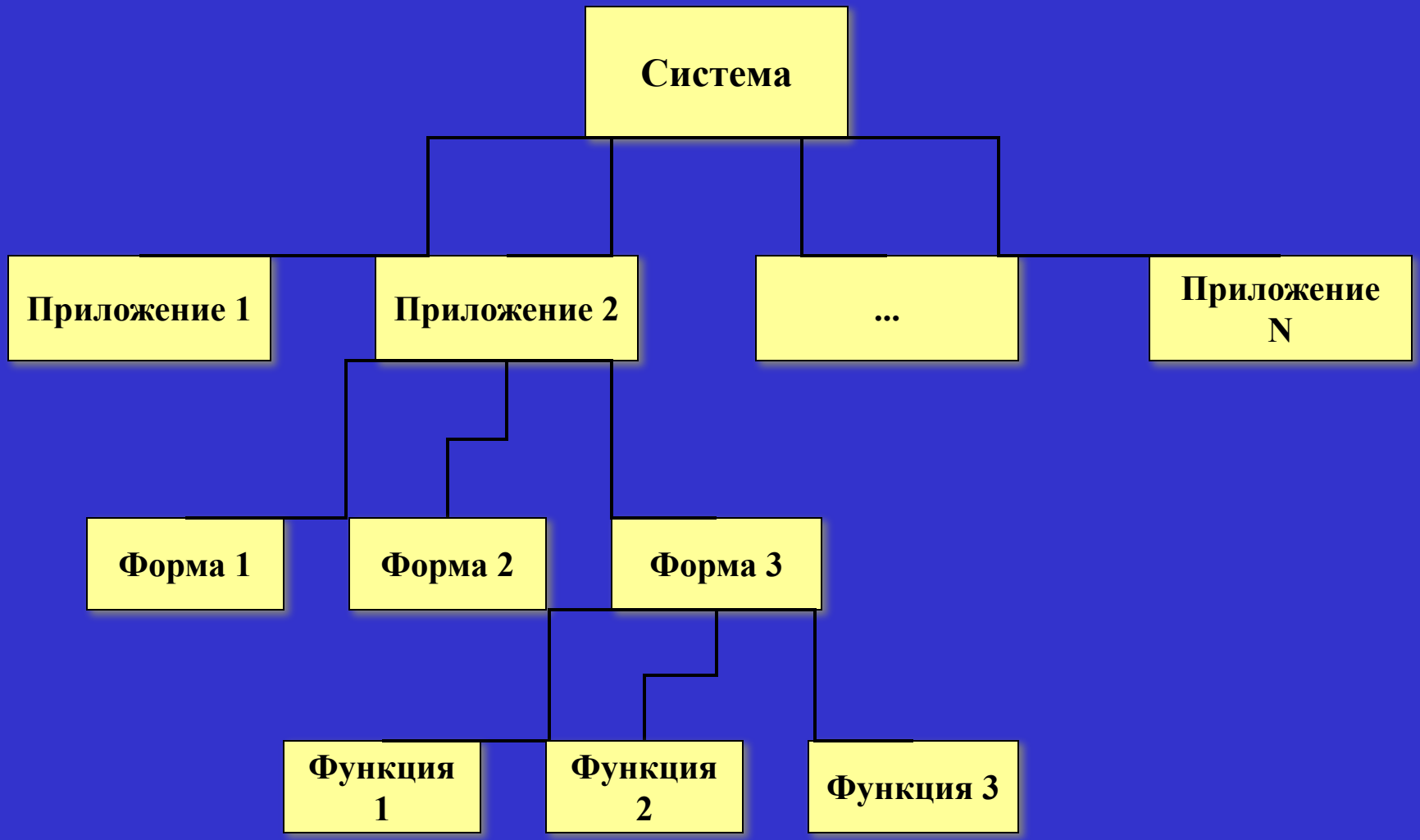
5. Модуль должен иметь один вход и один выход.

6. Модуль должен быть сравнительно небольшим (20 - 200 строк кода).

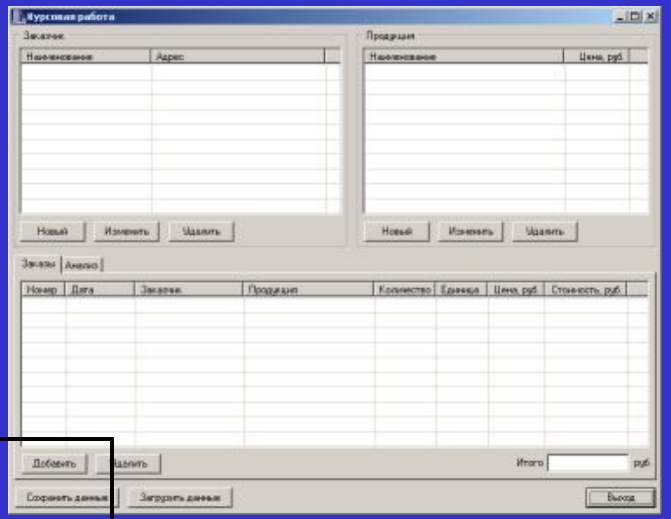
7. Модуль не должен быть зависим от истории своих вызовов.

8. В идеале модуль должен реализовывать одну функцию, причём целиком.

Модульность в среде Borland C++ Builder



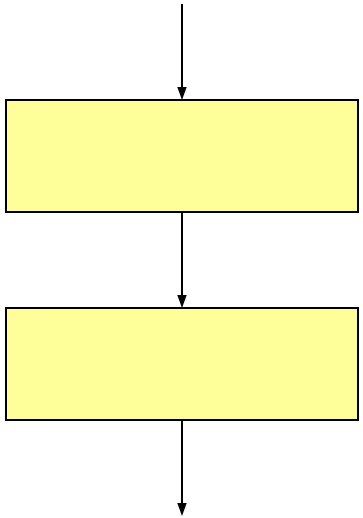
Пример нисходящей разработки приложения



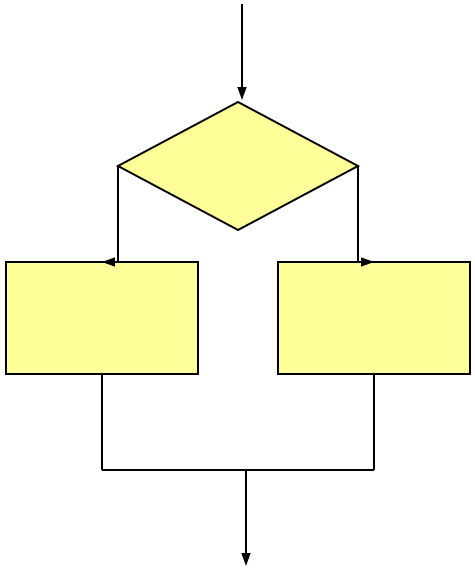
- **Управление заказами**
  - **Загрузка данных**
  - **Продукция**
    - **Создание**
    - **Редактирование**
    - **Поиск**
    - **Отображение**
    - **Удаление**
  - **Клиенты**
  - **Заказы**
  - **Сохранение данных**

Структурное программирование

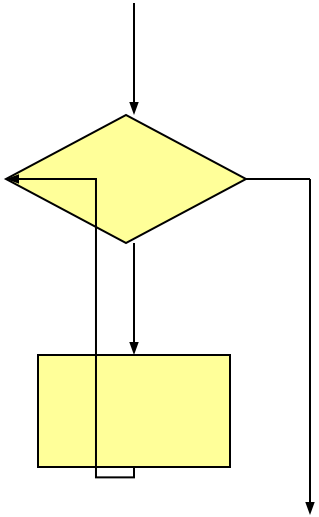
Следование



Развилка (условие)

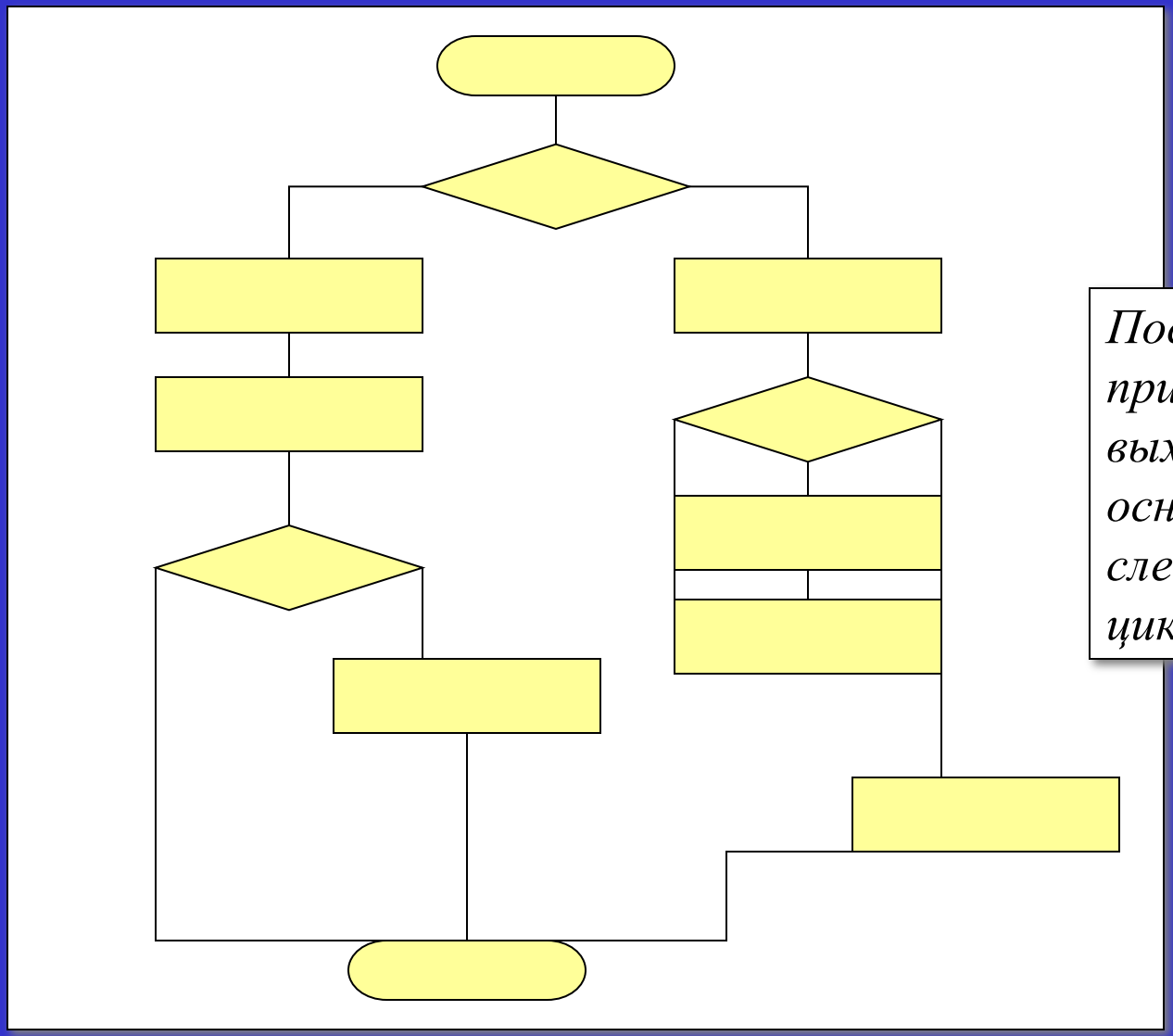


Цикл



Логическая структура программы может быть выражена комбинацией трех базовых структур

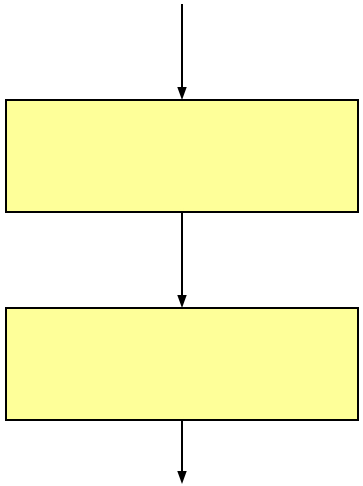
Комбинирование трёх конструкций



Построение модулей по принципу «один вход - один выход», комбинируя три основные конструкции: следования, развилки и цикла

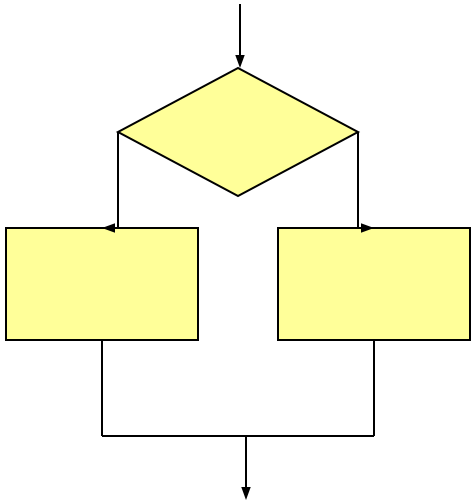
Псевдокод

Следование



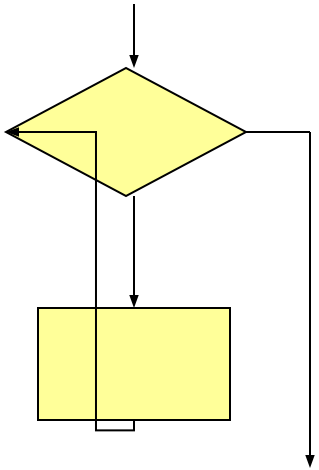
Действие 1
Действие 2
Действие 3

Развилка (условие)



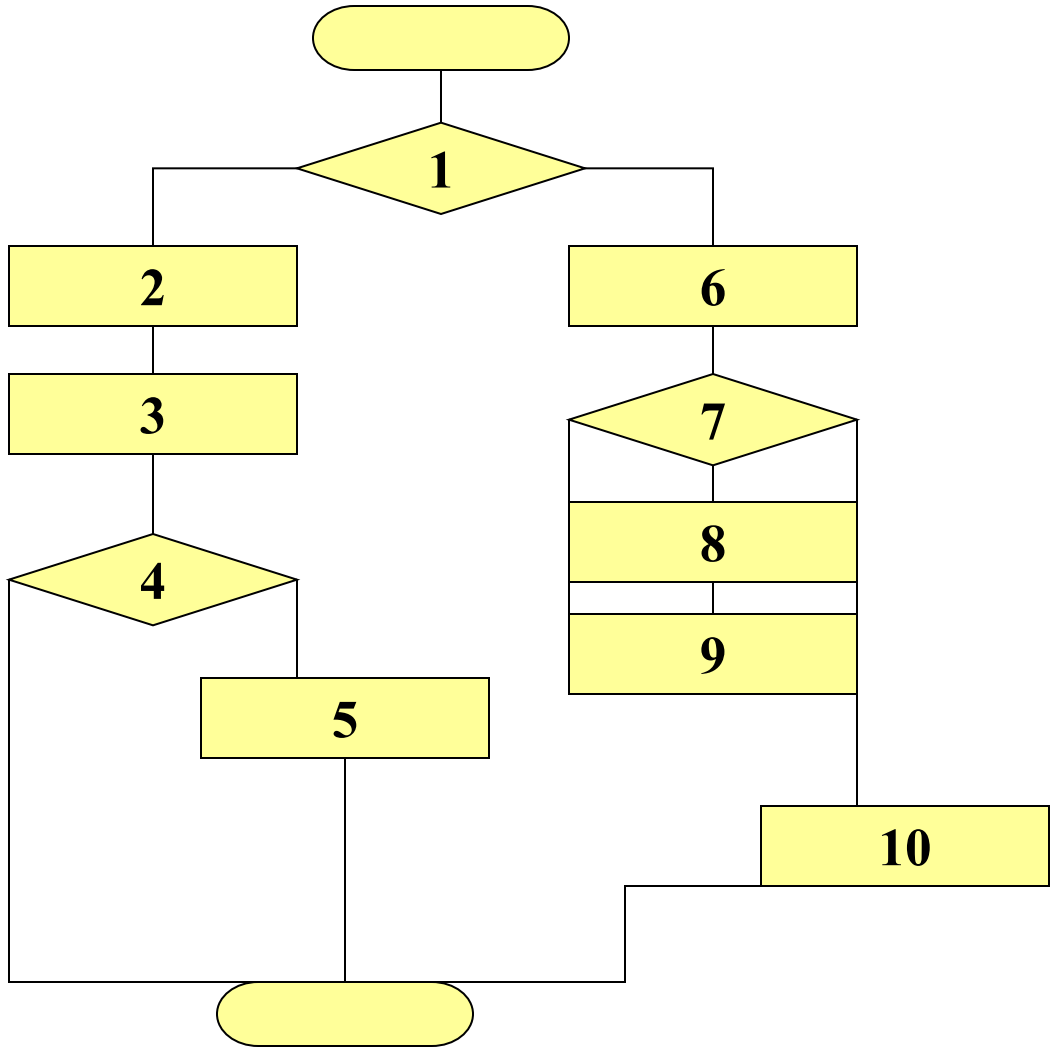
ЕСЛИ *условие*
ТО
    Действия
ИНАЧЕ
    Действия
ВСЁ-ЕСЛИ

Цикл



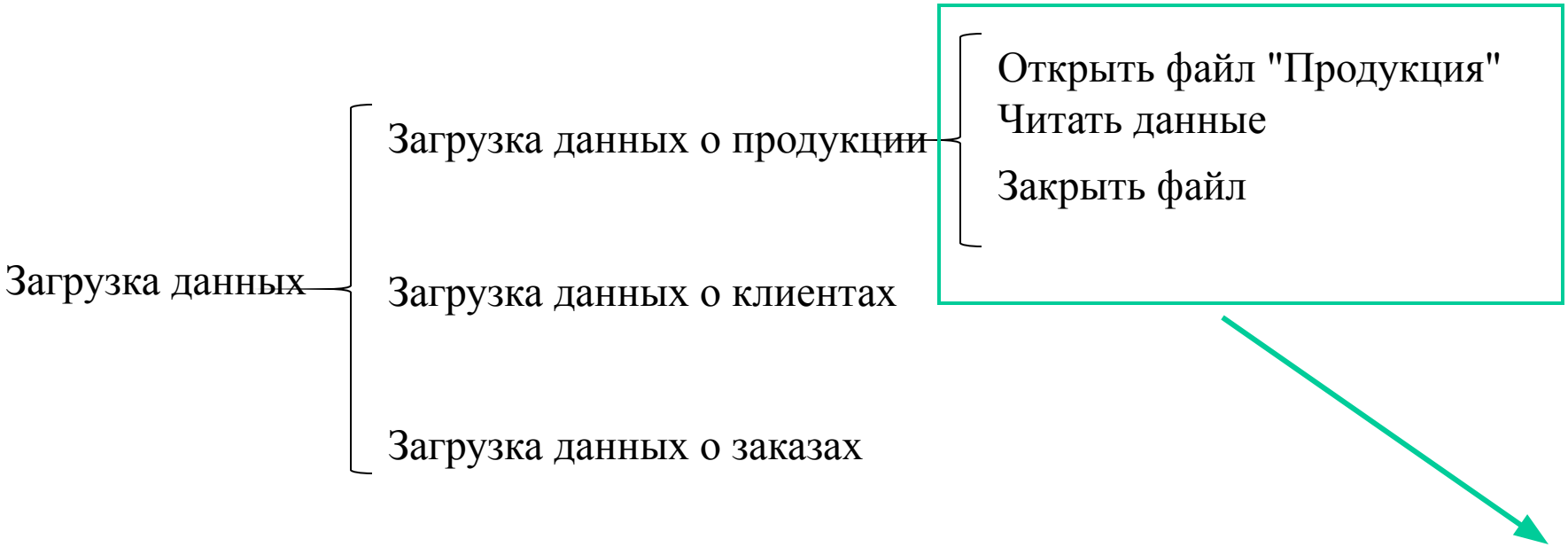
ЦИКЛ ПОКА *условие*
    Действия
ВСЁ-ЦИКЛ

Использование псевдокода



ЕСЛИ *условие 1*
ТО
    Действие 2
    Действие 3
    ЕСЛИ *условие 4*
    ТО
    ИНАЧЕ
        Действие 5
    ВСЁ-ЕСЛИ
ИНАЧЕ
    Действие 6
    ЦИКЛ-ПОКА *условие 7*
        Действие 8
        Действие 9
    ВСЁ-ЦИКЛ
    Действие 10
ВСЁ-ЕСЛИ

Пошаговая детализация



От пошаговой детализации к псевдокоду

```
Открыть файл "Продукция"
ЕСЛИ успешно
ТО
    Сбросить счетчик элементов массива "Продукция"
    ЦИКЛ-ПОКА не встречен конец файла
        Читать очередную запись в массив
        Увеличить счетчик на 1
    ВСЁ-ЦИКЛ
    Закрыть файл "Продукция"
ИНАЧЕ
    Сообщение об ошибке "Файл не найден"
ВСЁ-ЕСЛИ
```

От псевдокода к программе на C++

Открыть файл "Продукция"
ЕСЛИ успешно
ТО
    Сбросить счетчик элементов
    ЦИКЛ-ПОКА до конца файла
        Читать очередную запись
        Увеличить счетчик на 1
    ВСЁ-ЦИКЛ
    Закрыть файл "Продукция"
ИНАЧЕ
    Сообщение об ошибке
ВСЁ-ЕСЛИ

```
FILE* pf = fopen("product.dat", "rb");
if(pf)
{
   prod_count = 0;
   while(!feof(pf))
   {
      fread(&prod[prod_count],
            sizeof(PROD), 1, pf);
      prod_count++;
   }
   fclose(pf);
}
else
   ShowMessage("Файл не найден");
```

Оптимизация кода C++

```
if(FILE* pf = fopen("product.dat", "rb"))
{
   for(prod_count = 0; !feof(pf); prod_count++)
      fread(&prod[prod_count], sizeof(PROD), 1, pf);

   fclose(pf);
}
else
   ShowMessage("Файл не найден");
```

Некоторые приемы структурного программирования на C++

Макросы

```
#define MODULE(a, b) \
...
```

Функции

```
void MODULE(int a, int b)
{ ... }
```

Шаблоны

```
template<class x>
MODULE(x a, x b)
{ ... }
```

Однократное описание повторяющегося кода

Некоторые приемы структурного программирования на C++

```
int a;
int b;
int c;

void main()
{
   a = StrToInt(...);
   b = StrToInt(...);
   c = a+b;
}
```

```
void main()
{
   int a;
   int b;
   int c;

   a = StrToInt(...);
   b = StrToInt(...);
   c = a+b;
}
```

Отказ от глобальных переменных в пользу локальных

Некоторые приемы структурного программирования на C++

```
void main()
{
   int a;
   int b;
   int c;

   a = ...;
   b = ...;
   c = a+b;
}
```

```
void main()
{
   int a = ...;
   int b = ...;
   int c = a+b;
}
```

Возможно более позднее определение переменных

Некоторые приемы структурного программирования на C++

```
int i;

for(i = 0; i < N; i++)
   a += b[i];

for(i = 0; i < M; i++)
   a -= c[i];
```

```
for(int i = 0; i < N; i++)
   a += b[i];

for(int i = 0; i < M; i++)
   a -= c[i];
```

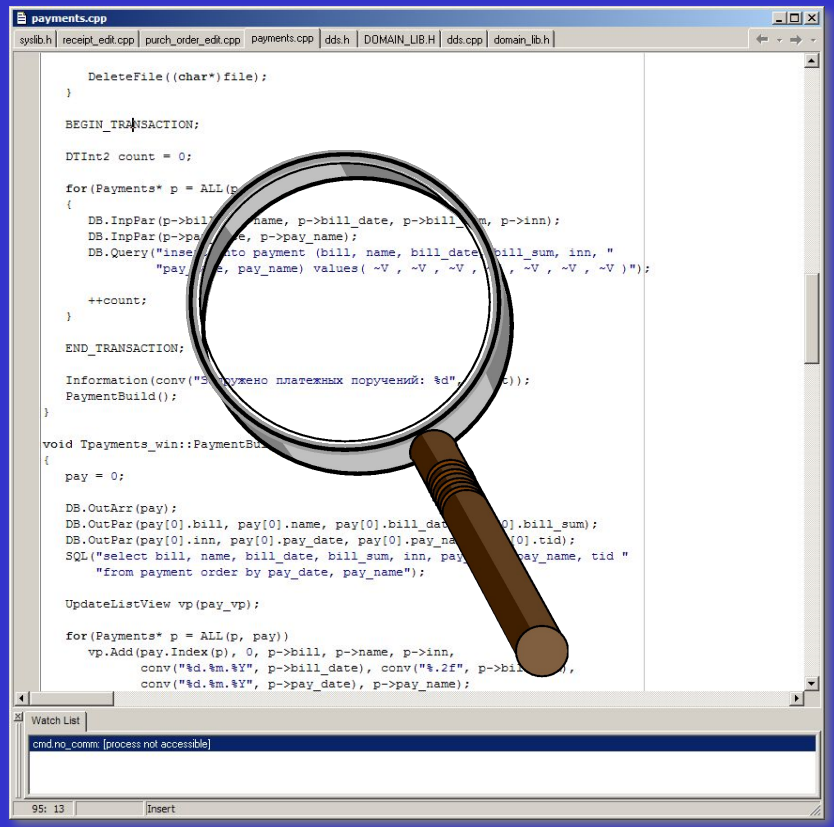
```
FILE* inp;
if(inp = fopen(...))
{
   ...
}
```

```
if(FILE* inp = fopen(...))
{
   ...
}
```

Сокращение области видимости переменных

Сквозной структурный контроль

Обнаружение и исправление ошибок на ранних стадиях проекта, пока стоимость исправления минимальна, а последствия наименее значительны



Верификация программного обеспечения

Проверка правильности работы программ

Peer review (проверка кода)

Тестирование (проверка результатов)