



**ПОЛИТЕХ**

Санкт-Петербургский  
политехнический университет  
Петра Великого



**ИКНТ**

# Верификация и анализ программ. Модели программ

2020



КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ПРОГРАММНЫЕ ТЕХНОЛОГИИ

# Модели программ

- ▶ Модели программ:
  - Структурные модели
  - Поведенческие модели
- ▶ Структурные модели основаны на
  - Синтаксисе исходного языка
  - Структуре программы
- ▶ Поведенческие модели
  - Дополнительно используют семантику конструкций языков программирования

# Модели программ

- ▶ Структурные модели
  - Дерево разбора (синтаксическое дерево, parsing tree, concrete syntax tree)
  - Абстрактное синтаксическое дерево (abstract syntax tree, AST)
- ▶ Поведенческие модели
  - Граф потока управления (Control Flow Graph, CFG)
  - Статическое однократное присваивание (Single Static Assignment, SSA)
  - Граф зависимостей по данным (Data Dependency Graph, DDG)
  - Граф зависимостей программы (Program Dependency Graph, PDG)
- ▶ Гибридные модели
  - Абстрактный семантический граф (Abstract Semantic Graph, ASG)

# Дерево разбора

- ▶ Синтаксическое дерево, конкретное синтаксическое дерево, parsing tree
- ▶ Является формой представления программы, соответствующей грамматике исходного языка (например, БНФ или РБНФ)
- ▶ Строится парсерами написанными вручную или синтезированными генераторами парсеров
- ▶ Является наиболее детальным структурным представлением программы
- ▶ В исходном виде редко используется для анализа программ из-за громоздкости

# Абстрактное синтаксическое дерево

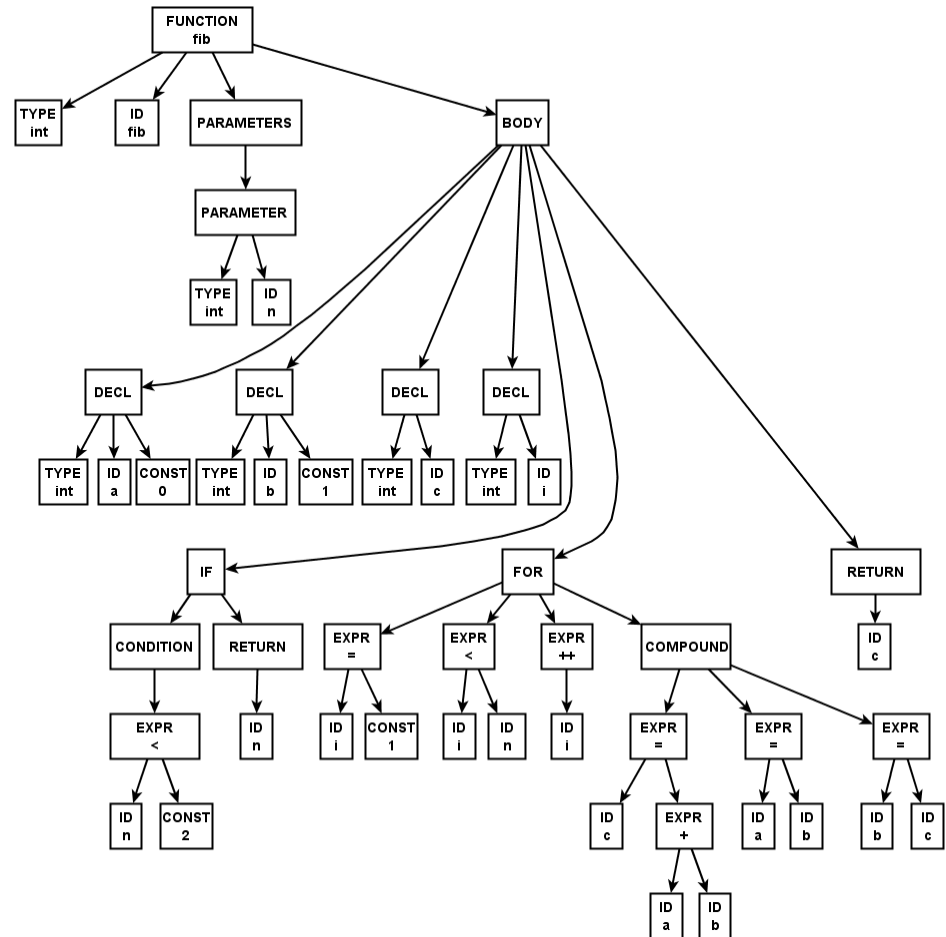
- ▶ Структурное представление программы
- ▶ Отличие от синтаксического дерева (дерева разбора):
  - Удаление большинства нетерминальных узлов, имеющих одного потомка
  - Замена части терминальных символов атрибутами узлов
  - Модификация группирующих узлов
- ▶ По модели АСД легко восстановить исходную программу с точностью до форматирования и комментариев
- ▶ Строится из дерева разбора путем несложной постобработки

# Абстрактное синтаксическое дерево

```

int fib(int n)
{
    int a = 0, b = 1, c, i;
    if (n < 2) return n;
    for (i = 1; i < n; i++)
    {
        c = a + b;
        a = b;
        b = c;
    }
    return c;
}

```



# Граф потока управления

- ▶ Граф потока управления (ГПУ), Control Flow Graph (CFG)
- ▶ Потоки управления программы представляются в виде направленного графа
- ▶ Определяется не только структурой программы, но и семантикой операторов
- ▶ Два узла графа (А и В) связаны направленной дугой, если возможно выполнение В непосредственно после А
- ▶ ГПУ используется для:
  - Генерации кода
  - Оптимизации программ
  - Обнаружения ошибок в программе
  - И т.п.

# Построение графа потока управления

- ▶ ГПУ строится с помощью анализа АСД на основе семантики языковых конструкций
  - Для большинства конструкций – дуга, соединяющая узлы, соответствующие следующим друг за другом операторам
  - Для части конструкций (ветвления, циклы, вызов функций, переходы и т.п.) – в соответствии с семантикой
- ▶ Проблемные конструкции языков программирования:
  - Вычисляемые безусловные переходы (goto)
  - Вызов функции по указателю
  - Полиморфизм
  - Структурная обработка исключений
  - Параллелизм
  - И т.п.

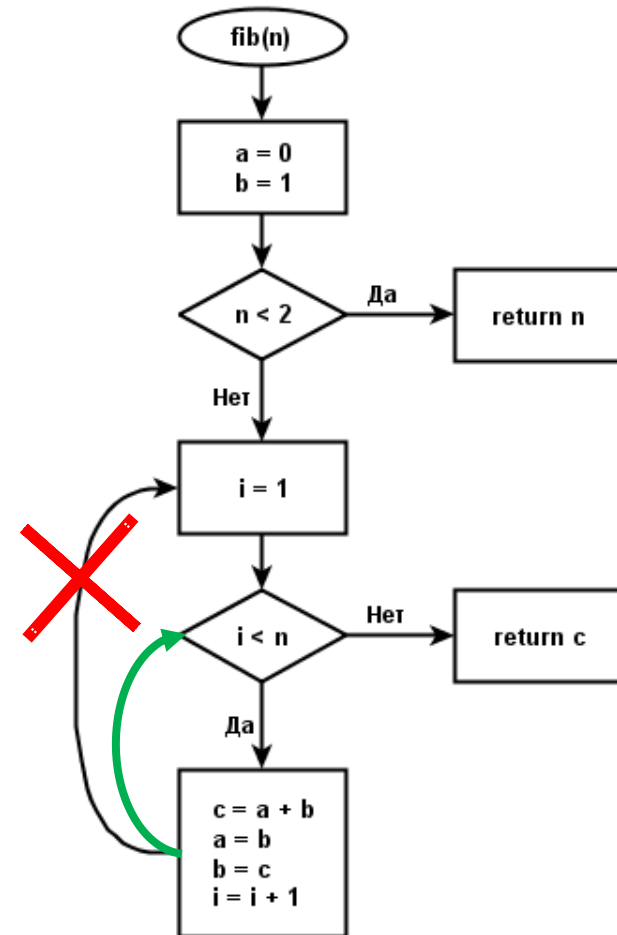


# Граф потока управления

```

int fib(int n)
{
    int a = 0, b = 1, c, i;
    if (n < 2) return n;
    for (i = 1; i < n; i++)
    {
        c = a + b;
        a = b;
        b = c;
    }
    return c;
}

```



# Граф зависимостей по данным

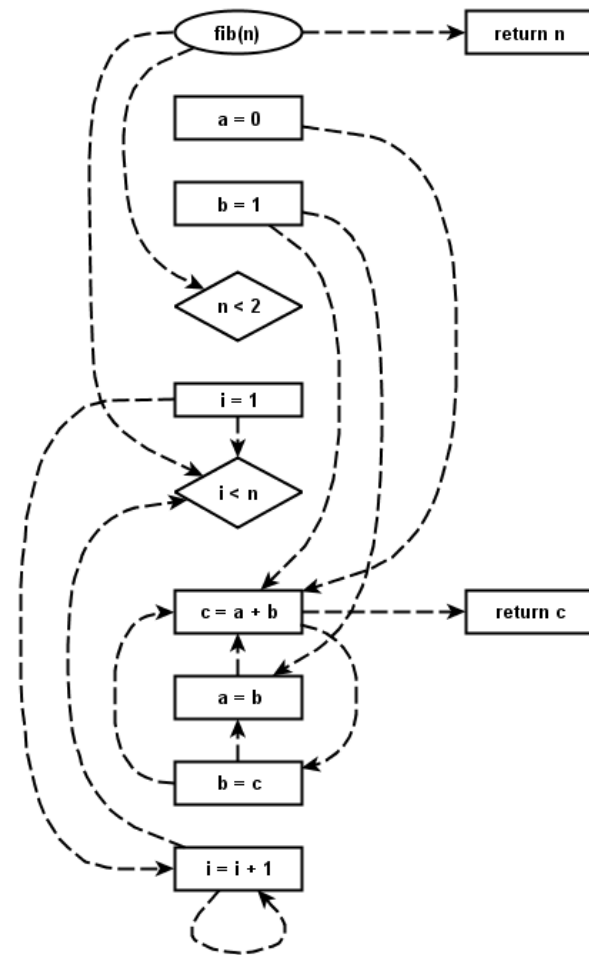
- ▶ Граф зависимости по данным (ГЗД), Data Dependency Graph (DDG)
- ▶ Отображает связи по данными между объектами программы
- ▶ Показывает зависимости между объявлениями, инициализациями, присваиваниями объектов и их использованиями
- ▶ Определяется структурой программы, семантикой конструкций
- ▶ Используется для:
  - Оптимизации программ
  - Автоматического распараллеливания программ
  - Обнаружения неинициализированных объектов
  - Обнаружения некорректно инициализированных объектов
  - и т.п.

# Граф зависимостей по данным

```

int fib(int n)
{
    int a = 0, b = 1, c, i;
    if (n < 2) return n;
    for (i = 1; i < n; i++)
    {
        c = a + b;
        a = b;
        b = c;
    }
    return c;
}

```



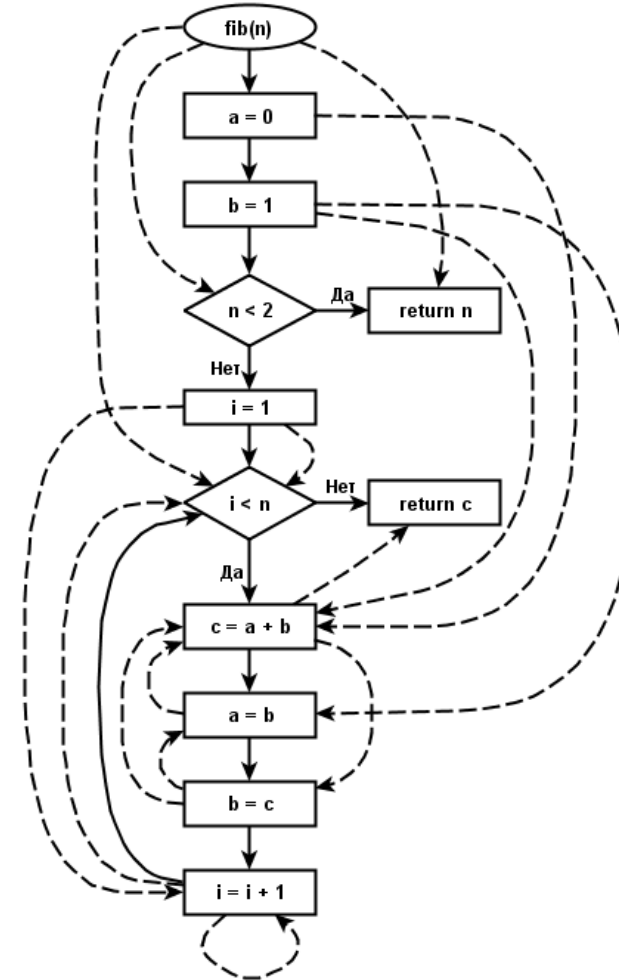
# Граф зависимостей программы

- ▶ Является комбинированной моделью
- ▶ Содержит два типа связей:
  - Связи потока управления
  - Зависимости по данным
- ▶ Используется для
  - Оптимизации
  - Генерации кода
  - Анализа программ
  - Верификации
  - И т.п.

# Граф зависимостей программы

```

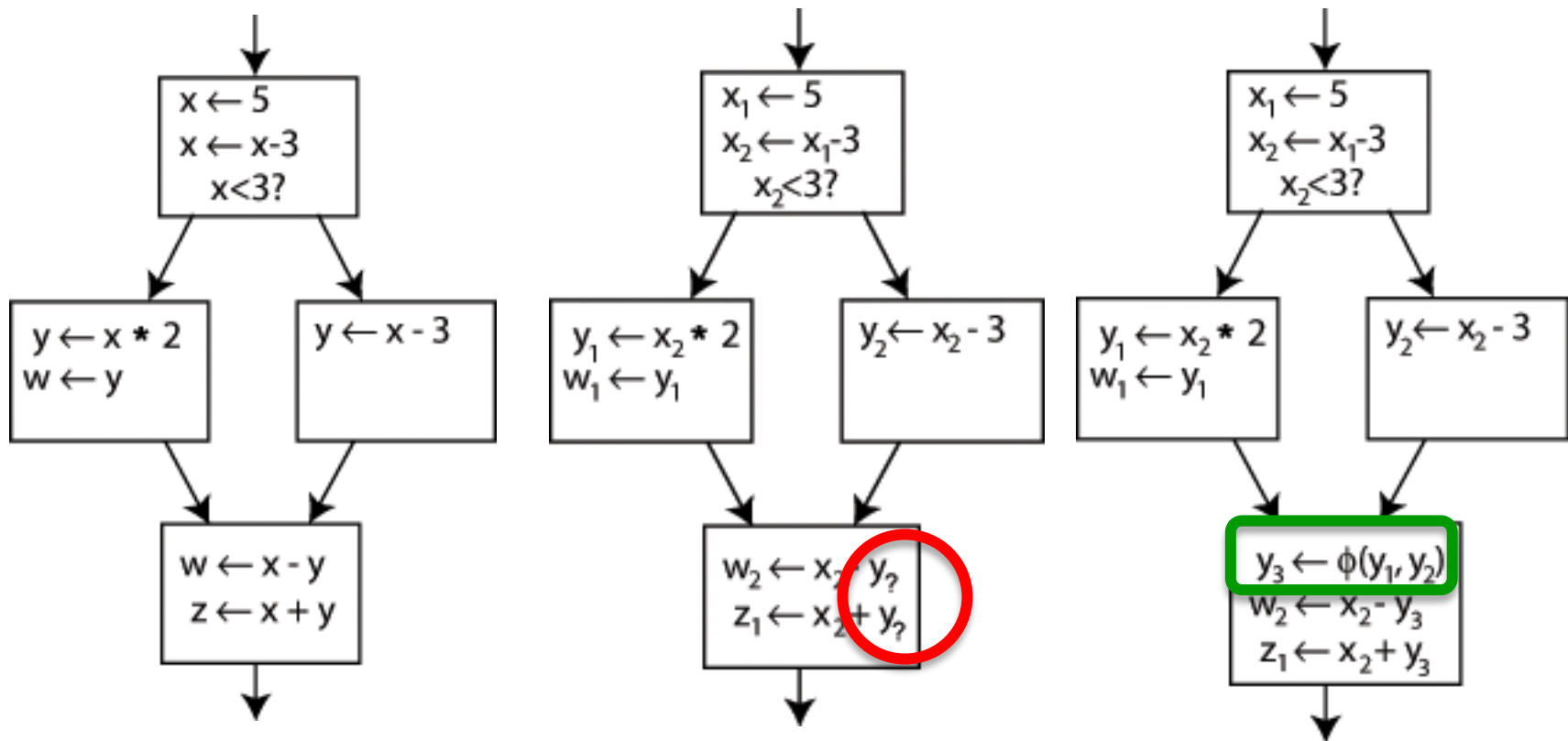
int fib(int n)
{
    int a = 0, b = 1, c, i;
    if (n < 2) return n;
    for (i = 1; i < n; i++)
    {
        c = a + b;
        a = b;
        b = c;
    }
    return c;
}
  
```



# Статическое однократное присваивание

- ▶ Статическое однократное присваивание, Static Single Assignment form(SSA)
- ▶ Разработано IBM в 1981 году
- ▶ Модель, объединяющая граф потока управления и элементы зависимости по данным
- ▶ Значения всем объектам программы присваиваются один раз
- ▶ Вводится версионирование для каждого очередного присваивания объекта
- ▶ При обращении к значению объекта используется последняя версия по ходу графа потока управления
- ▶ Неоднозначность разрешается с помощью  $\phi$ -функции

# Статическое однократное присваивание



# Статическое однократное присваивание

- ▶ Используется в компиляторах, статических анализаторах, верификаторах:
  - Распространение констант (constant propagation)
  - Обнаружение и удаление мёртвого кода
  - Обнаружение неинициализированных объектов
  - Распараллеливание
  - Генерация кода
  - Преобразование программы / части программы в формулы для последующей верификации
  - И т.п.

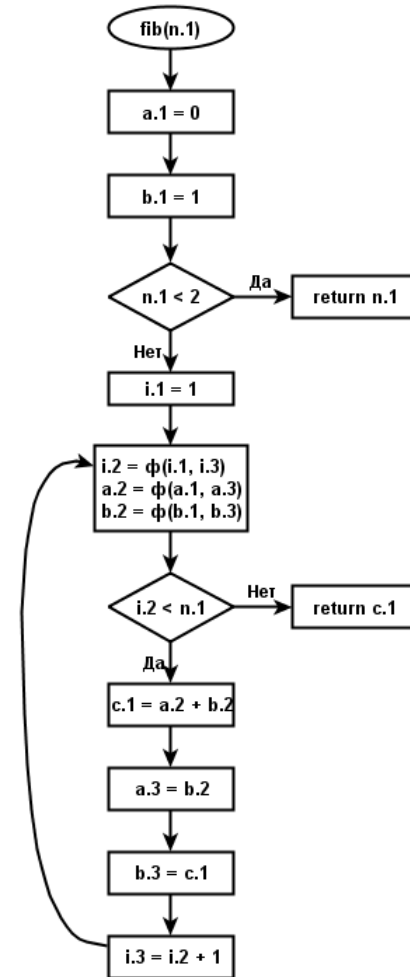


# Статическое однократное присваивание

```

int fib(int n)
{
    int a = 0, b = 1, c, i;
    if (n < 2) return n;
    for (i = 1; i < n; i++)
    {
        c = a + b;
        a = b;
        b = c;
    }
    return c;
}

```



# Абстрактный семантический граф

- ▶ Наиболее общая модель, объединяющая структурные и поведенческие компоненты
- ▶ Абстрактный синтаксический граф обогащается семантической информацией:
  - Связи по данным
  - Связи по управлению
  - Типами данных
  - И т.п.

# Абстрактный семантический граф

```

int fib(int n)
{
  int a = 0, b = 1, c, i;
  if (n < 2) return n;
  for (i = 1; i < n; i++)
  {
    c = a + b;
    a = b;
    b = c;
  }
  return c;
}

```

