ПРАКТИКУМ НА ЭВМ
Задание 1:
Оценка числа состояний программы

ВМК МГУ им. М.В. Ломоносова,
Кафедра АСВК
к.ф.-м.н., доцент Волканов Д.Ю.
аспирант Степанов Е.П.

Практикум

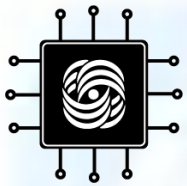
- 6 заданий, по мере прохождения курса
- Первое задание – 01.10
- Умение проектировать и писать на C++
- Каждое задание - 10 баллов
- Бонусы за время
- Каждая неделя сверх

Дедлайна –N задания

Оценка за Практикум

- 60 баллов – отлично
- 50 баллов – хорошо
- 40 баллов – удовлетворительно
- На комиссии ставится ладно
- Временной бонус:
  - +50% на 1й неделе
  - +20% на 2й неделе
  - Далее без бонуса

  на 3й и 4й неделях



Моделирование программ

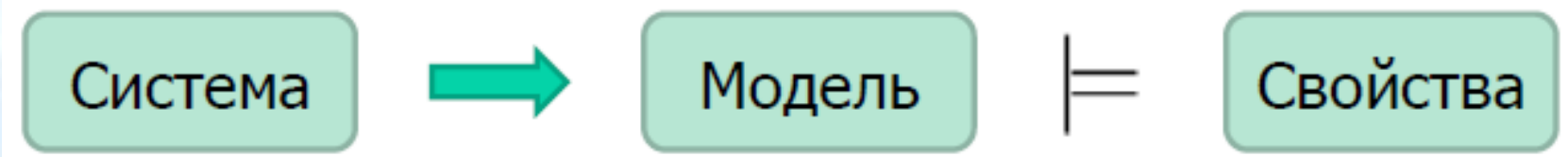
(пример построения модели,
общая схема моделирования)

Зачем строить модели программ?

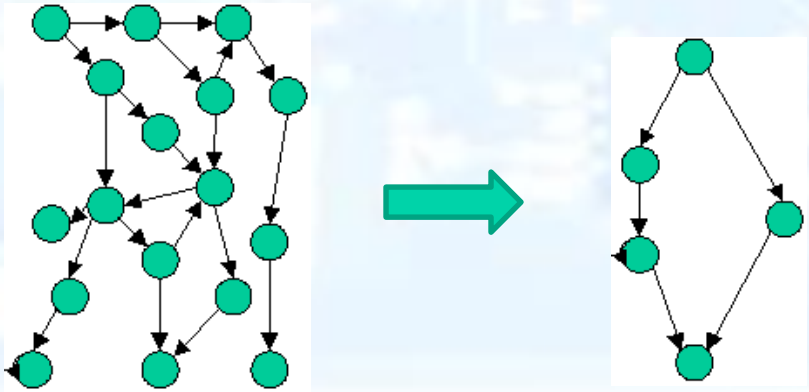
- Свойства задают допустимые состояния и последовательности состояний (вычисления)

т.н. свойства линейного времени

- Модель генерирует трассы, по которым можно проверить свойства,



- Абстракция от лишних состояний.



Состояние программы

- Состояние программы – совокупность значений объектов данных и счётчика управления,
- Как правильно оценить количество состояний?

Спецификация программы	**Исходный код на языке высокого уровня**	**Исполняемый код (ASM)**	**Программно - аппаратная система**	**Работающий компьютер**
Может вообще не быть состояний (stateless протоколы)	Данные – переменные языка, счетчик – номер выполняемого оператора	Данные – ячейки памяти в адресном пр-ве процессора и регистры, счётчик -- регистр команд процессора	Данные – ячейки памяти различных ЗУ, кэш, буфера выборки, состояние конвейеров, Счётчик – адрес последней команды	Атомы, электроны, спины …

Состояние программы (на уровне языка С)

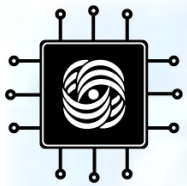
```
    int compare(int a, int b) {
0:      int res;
1:      if (a < b) {
2:          res = -1;
3:      } else if (a >= b) {
4:          res = 1;
        } else {
5:          res = 0;
        }

6:      return res;
7:  }
```

Потенциально –

$8*2^{96}$

состояний

3 целочисленных переменных,
$|int| = 2^{32}$,
7 операторов языка +
терминальный оператор.



Состояние программы (на уровне языка C)

```
    void foo(int a) {
0:      int  x= 5;
1:      if   (a < x) {
2:      x =  x-a;
3:      }
4:  }

    void bar(int b) {
0:        int x;
1:        x = b;
2:  }
```

Потенциально –

$15*2^{128}$

состояний

4 целочисленных переменных, $|int| = 2^{32}$,
2 потока управления, в foo – 5 операторов, в bar – 3.

Достижимые состояния

- Состояние называется **достижимым**, если существует вычисление программы, в котором оно присутствует
  - формализация – позже
- Достижимость состояния в программе в общем случае алгоритмически неразрешима.
  - неразрешима даже достижимость точки программы

Число достижимых состояний

```
    int compare(int a, int b) {
0:          int res;
1:          if (a < b) {
2:                  res = -1;
3:          } else if (a >= b) {
4:                  res = 1;
            } else {
5:                  res = 0;
            }

6:          return res;
7:  }
```

$8*4*2^{64}$

переменная `res` может принимать только 4 значения

$7*3*2^{64}$

5-й оператор не достижим

Пример построения модели программы

- Программа подсчёта количества слов в файле
- Проверяемое свойство – всегда ли программа закрывает открытый файл?
- См. следующий слайд.

Пример построения модели программы

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc,
             const char* argv[]) {
  FILE* f;
  int c, wordCnt = 0, inWord = 0;
  if (argc < 2) {
    printf("Use: %s filename\n",
                            argv[0]);
    return 1;
  }
  f = fopen(argv[1], "r");
  if (f == NULL) {
    printf("Can't open file:
                        %s\n", argv
                        [1]);
    return 1;
  }
```

```
while ((c = fgetc(f)) != -1) {
    if (!isspace(c)) {
      if (!inWord) {
        ++wordCnt;
        inWord = 1;
      }
    } else {
          inWord = 0;
    }
  }
  printf("Word count: %d\n",
                        wordCnt);
  fclose(f);
  return 0;
}
```

Пример построения модели программы

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc,
             const char* argv[]) {
  FILE* f;
  int c, wordCnt = 0, inWord = 0;
  if (argc < 2) {
    printf("Use: %s filename\n",
                              argv[0]);
    return 1;
  }
  f = fopen(argv[1], "r");
  if (f == NULL) {
    printf("Can't open file:
                        %s\n", argv
                        [1]);
    return 1;
  }
```

```
while ((c = fgetc(f)) != -1) {
    if (!isspace(c)) {
      if (!inWord) {
        ++wordCnt;
        inWord = 1;
      }
    } else {
          inWord = 0;
    }
  }
  printf("Word count: %d\n",
                        wordCnt);
  fclose(f);
  return 0;
}
```

Пример построения модели программы

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  FILE* f;
  int c, wordCnt = 0, inWord = 0;
  enum {FALSE, TRUE} P1 = any();
  if (P1) {
    return 1;
  }
  f = fopen("sample", "r");
  if (f == NULL) {
    return 1;
  }
```

```
while ((c = fgetc(f)) != -1) {
    if (!isspace(c)) {
      if (!inWord) {
        ++wordCnt;
        inWord = 1;
      }
    } else {
            inWord = 0;
    }
  }
  fclose(f);
  return 0;
}
```

Пример построения модели программы

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  FILE* f;
  int c, wordCnt = 0, inWord = 0;
  enum {FALSE, TRUE} P1 = any();
  if (P1) {
    return 1;
  }
  f = fopen("sample", "r");
  if (f == NULL) {
    return 1;
  }
```

```
while ((c = fgetc(f)) != -1) {
    if (!isspace(c)) {
      if (!inWord) {
        ++wordCnt;
        inWord = 1;
      }
    } else {
          inWord = 0;
    }
  }
  fclose(f);
  return 0;
}
```

Пример построения модели программы

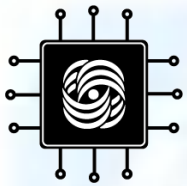
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  FILE* f;
  int inWord = 0;
  enum {FALSE, TRUE} P1 = any(),
                     P2, P3;
  if (P1) {
    return 1;
  }
  f = fopen("sample", "r");
  if (f == NULL) {
    return 1;
  }
```

```
while (P2 = any()) {
    if (P3 = any()) {
      if(inWord)
        inWord = 1;
      }
    } else {
          inWord = 0;
    }
  }
  fclose(f);
  return 0;
}
```

Пример построения модели программы

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  FILE* f;
  int inWord = 0;
  enum {FALSE, TRUE} P1 = any(),
                     P2, P3;
  if (P1) {
    return 1;
  }
  f = fopen("sample", "r");
  if (f == NULL) {
    return 1;
  }
```

```
while (P2 = any()) {
    if (P3 = any()) {
      if(inWord)
        inWord = 1;
      }
    } else {
            inWord = 0;
    }
  }
  fclose(f);
  return 0;
}
```



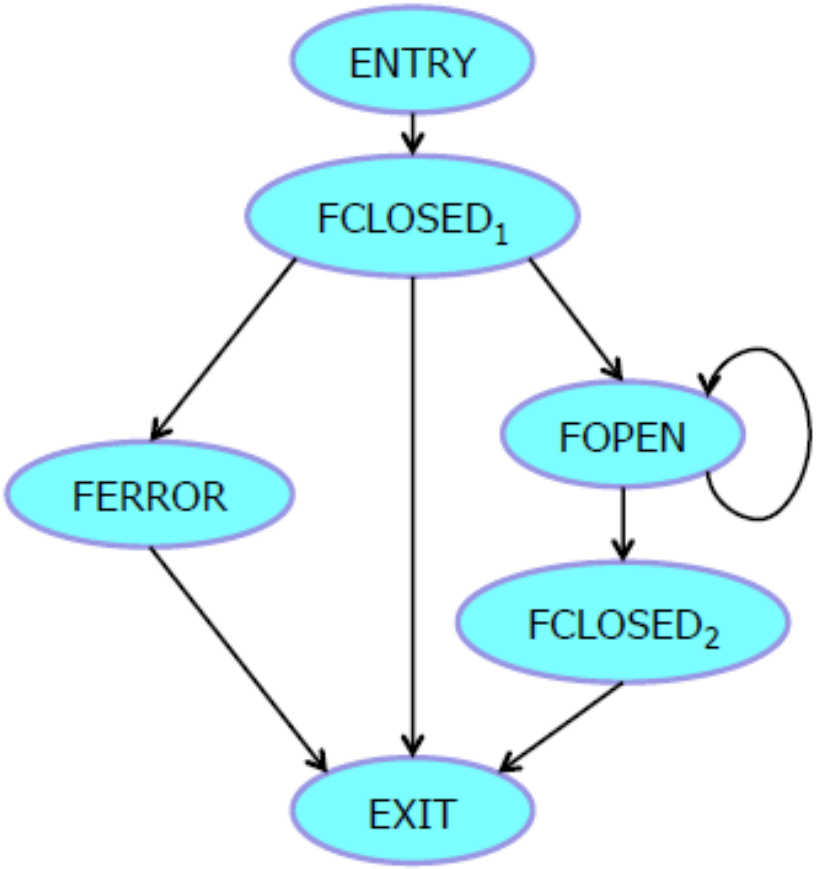
Пример построения модели программы

```
int main() {
  enum { FCLOSED, FOPEN, FERROR} fileState;
  enum { V0, V1 } inWord = V0;
  enum {FALSE, TRUE} P1 = any(),
                     P2, P3;
  if (P1) {
    return 1;
  }
  if(any()) {
    fileState = FOPEN;
  } else {
    fileState = FERROR;
  }
  if (fileState == FERROR) {
    return 1;
  }
```

```
while (P2 = any()) {
    if (P3 = any()) {
      if (!inWord) {
        inWord = V1;
      }
    } else {
        inWord = V0;
    }
  }
  fileState = FCLOSED;
  return 0;
}
```

Пример построения модели программы

```
int main() {
  enum { FCLOSED, FOPEN, FERROR} fileState;
  if(any()) {
    fileState = FERROR;
  } else if (any()) {
    fileState = FOPEN;
    while (any());
    fileState = FCLOSED;
  }
  return 0;
}
```



Дано

- Текст программы на языке C с функциями f и g.

Предположим

- Функции f и g выполняются параллельно, в двух разных потоках управления (процессах) P_f и P_g соответственно.
- Будем считать состоянием модели программы совокупность значений счётчиков команд c_f и c_g процессов P_f и P_g, значения глобальных и локальных переменных заданной программы.
- Неинициализированные переменные принимают специальное значение, обозначаемое символом #, лежащее вне диапазона MININT..MAXINT.

Требуется

- (1 балл) Оценить размер множества потенциальных состояний программы. Ответ обосновать.
- (2 балла) Оценить размер множества достижимых состояний программы. Ответ обосновать. В обосновании описать множество достижимых состояний с помощью линейных равенств и неравенств на языке C.

Например, (c_f == 3 && a < 5) || (c_g == 4 && b > d).

Требуется

- (7 баллов) Написать программу на языке C++, которая вычисляет и записывает в текстовой файл states.txt множество достижимых состояний заданной программы для заданных значений параметров функций, а также выводит в стандартный поток вывода количество достижимых состояний.

Требования к файлам решения:

- Исходная формулировка задачи с именем task.txt
- Описание решения пп. 1-2 в текстовом файле с именем solution.txt. Текстовый файл должен быть в одной из кодировок: utf8, cp1251 (Windows), koi8-r. Файлы других форматов (doc, pdf, etc) не принимаются.
- Файл group_surname.cpp с программой подсчёта состояний (group - номер вашей группы, surname - ваша фамилия латиницей).

Требования к программе (1)

- 1. Программа должны вычислять множество достижимых состояний программы, присланной вам на почту, для заданных значений параметров функций f и g.

- 2. Программа должна уметь выводить множество состояний в текстовой файл в формате: значение счётчика f, значение счётчика g, значение h, значение f.x, значение f.y, значение g.x, значение g.y, (через запятую) по одному состоянию в строке. Значения счётчиков нумеруются с 0. Первая строка файла должна содержать описание формата вывода, а именно: c_f, c_g, h, f.x, f.y, g.x, g.y. Обратите внимание, что смена состояния программы происходит после выполнения соответствующего оператора программы (например, присваивания).

Требования к программе (2)

- 3. Программа должны выполняться в консоли. Обязательными входными параметрами являются значения параметров функций a и b в следующем порядке:

имя_программы <f_a> <f_b> <g_a> <g_b>

Также должны поддерживаться параметры:

- -file имя_файла - запись состояний в указанный файл,
- -count - вывод общего количества состояний программы в стандартный поток вывода.

При запуске без параметров (либо с недостаточным количеством параметров) программа должна выводить информацию о программе, авторе, годе написания и параметрах запуска.

Требования к программе (3)

- 4. Исходный код программы должен содержать информацию об авторе и годе написания программы.
- 5. Программа должна быть написана на языке ANSI/ISO C++.
- 6. Программа должна успешно компилироваться компилятором gcc со следующими параметрами: g++ -O2 -Wall -Werror.
- 7. Допускается использование только библиотек stdlib и STL.

Требования к программе (4)

- 8. Текст программы должен быть снабжён исчерпывающими комментариями. Как минимум, должны быть прокомментированы все объявления функций, операторы ветвления и линейные участки программы.

- 9. Программа должна быть написана самостоятельно. При заимствовании фрагментов кода из open-source проектов должен быть указан автор кода. Заимствовать код из решений Практикумов прошлых лет не допускается. Если будет обнаружено, что несколько присланных решений написаны одним человеком, оценка будет снижена для всех похожих работ. Для анализа схожести кода будут использованы соответствующие инструментальные средства.

Штрафы

- Неправильная тема письма (-1)
- Нет task.txt (-1)
- Нет solution.txt (-3)
- Нет обоснования числа потенциальных состояний (-1)
- Нет обоснования достижимых состояний (-2)
- Нет разметки операторов заданной программы номерами (-1)
- Код своей программы не откомментирован (-1)
- Состояние меняется до выполнения оператора (-2)
- Неправильные входные параметры или формат вывода (-2)
- Отсутствуют терминальные состояния (-2)
- Дублирующие состояния (-2)
- Потерянные состояния (-3)

Сроки

- Для получения задания прислать письмо с темой Praktikum-Task1-Request на volkanov@lvk.cs.msu.su
- Задача сдаётся в МЗ 758 14:35
  - 22.10 (+50%)
  - 29.10 (+20%)
  - 05.11, 12.11 (+0%)

- **Сданная задача** присылается на volkanov@lvk.cs.msu.su письмом с темой Praktikum-Task1-Solution
- Присланная до 23:59 (московское время) 12 ноября задача принимается (**БЕЗ ШТРАФА**), далее –1 балл за каждую неделю.
- В ***исключительных случаях*** возможна сдача по почте.

Спасибо за внимание