Задание 1. Потенциальные и достижимые состояния программы (50 баллов).

Формулировка задания

Дано:

Дана текст программы на языке C с функциями f и g.

Предполагается, что 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. **(5 баллов)** Оценить размер множества потенциальных состояний программы. Ответ обосновать.

2. **(10 баллов)** 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).

3. **(35 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.

На выполнение задания отводится две недели.

Требования к файлам решения:

Решение задачи должно включать в себя 3 файла:

1. Исходная формулировка задачи с именем task.txt
2. 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.**
3. Файл group_surname.c с программой подсчёта состояний (group - номер вашей группы, surname - ваша фамилия латиницей).

Требования к программе:

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 a.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).

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.

4. Исходный код программы должен содержать информацию об авторе и годе написания программы.

5. Программа должна быть написана на языке ANSI C либо ANSI/ISO C++.

6. Программа должна успешно компилироваться компилятором gcc со следующими параметрами: gcc -O2 -Wall -Werror -Wformat-security -Wignored-qualifiers -Winit-self -Wswitch-default -Wfloat-equal -Wshadow -Wpointer-arith -Wtype-limits -Wempty-body -Wlogical-op -Wstrict-prototypes -Wold-style-declaration -Wold-style-definition -Wmissing-parameter-type -Wmissing-field-initializers -Wnested-externs -Wno-pointer-sign.

7. Допускается использование только библиотек stdlib и STL.

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 предыдущего года 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.

Чек-лист и штрафы

Стандартные ошибки	**штраф**
Нет task.txt	-2
Нет solution.txt	-15
Нет обоснования числа потенциальных состояний	-3
Нет обоснования числа достижимых состояний	-10
Нет разметки операторов заданной программы	-2

номерами	
Код своей программы не откомментирован	-5
Состояние меняется до выполнения оператора	-10
Неправильные входные параметры или формат вывода	-10
Отсутствуют терминальные состояния	-10
Дублирующиеся состояния	-10
Потерянные состояния	-15

Методические указания по выполнению задания:

1. Оценка числа потенциальных состояний программы. При оценке количества потенциальных состояний программы оценивается мощность множества состояний данной программы. Поток управления программы при этом не учитывается. Таким образом, необходимо оценить мощность множеств значений счётчиков управления двух процессов, множеств возможных значений их параметров и локальных переменных, а также разделяемой переменной. Далее требуется оценить мощность множества потенциальных состояний программы в соответствии с определением, данным в формулировке задачи.

Пример

Пусть нам дана следующая программа:

```
int h;
void f(int a)
{
  h = a;
}

void g()
{
  int x;
  x = h;
 }
```

Пронумеруем её строки следующим образом:

```
        int h;
        void f(int a)
        {
1:       h = a;
2:       if(h < a) {
3:         h = 0;
         }
4:      }

        void g()
        {
1:        int x;
2:        x = h;
3:       }
```

Приняв мощность множества значений типа int за 2^{32}, мы получим оценку числа потенциальных состояний в $4*3*2^{96} = 12*2^{96}$.

2. Оценка числа достижимых состояний программы. Для оценки числа достижимых состояний программы необходимо оценить ограничения, накладываемые на множество потенциальных состояний потоком управления программы. Ограничения на множество возможных значений счётчика управления проистекают из недостижимости отдельных операторов программы. Ограничения на множество возможных значений переменных программы проистекает из набора операторов присваивания, присутствующих в программе, и их достижимости.

Рекомендуется следующая последовательность действий:

1) исходя из присутствующих в программе операторов присваивания, более точно оценить множества допустимых значений переменных программы.

2) для каждой точки программы, начиная с начальной, оценить возможные значения параметров, локальных и разделяемых переменных в данной точке, описав диапазоны значений при помощи линейных неравенств. Это позволит оценить количество возможных различных состояний для каждой точки программы.

3) исходя из полученных данных, выявить недостижимые операторы программы.

4) подкорректировать построенные линейные неравенства с учётом недостижимости отдельных операторов.

5) сложить полученные оценки для отдельных точек программы. При этом рекомендуется выполнять генерализацию с целью упрощения записи количества состояний, получая не точную оценку, а оценку сверху.

Пример

Оценим множества допустимых значений переменных программы из предыдущего примера.

1. $-2^{16} <= a <= 2^{16}$ (входной параметр типа int),
2. h = {#, a, 0}
3. x = {#, h}

Разметим программу из предыдущего примера линейными неравенствами:

```
        int h;
        void f(int a)
        {
1:       h = a;          //h = #, -2^16 <= a <= 2^16, x = #
2:       if(h < a) {     //h = a, -2^16 <= a <= 2^16, x = {#, a}
3:         h = 0;        //недостижимо
         }
4:      }                //h = a, -2^16 <= a <= 2^16, x = {#, a}

        void g()
        {
1:        int x;         // h = {#, a}, -2^16 <= a <= 2^16, x = #
2:        x = h;         // h = {#, a}, -2^16 <= a <= 2^16, x = #
3:      }                // h = {#, a}, -2^16 <= a <= 2^16, x = {#,a}
```

Таким образом, число достижимых состояний программы можно оценить сверху числом $36*2^{32} = 2*2*3*3*2^{32}$ (мощность мн-ва допустимых значений h * мощность мн-ва допустимых значений x * количество достижимых операторов f * количество достижимых значений g * мощность мн-ва допустимых значений параметра a).

3. Программа, вычисляющая количество достижимых состояний.

- Изменение состояния программы происходит после выполнения оператора, вызвавшего данное изменение,
- В частности, это потребует введения т.н. терминальных состояний, в которые программа переходит после выполнения последнего оператора.
- Помните, что в множестве нет одинаковых элементов, и множество состояний – не исключение.

Варианты для самостоятельного решения

Вариант 1.1.

```
int h;
void f(int a, int b)
{
    int x, y;
    x = 3;
    y = 5;
    h = b;
    if (h > b - x) {
        x = 4;
    }
    if (x > 7) {
        x = 3;
    }
    if (x > 1) {
        y = 1;
    }
    h = y;
}

void g(int a, int b)
{
    int x, y;
    x = 8;
    y = 2;
    h = a;
    x = 3;
    h = x + b;
    if (y > 5) {
        y = 1;
    }
    y = 1;
    while (x > 4) {
        if (h > 0)
            break;
        if (x > 10) {
            h = a + b;
            h = b + x;
        }
        y = 4;
    }
}
```

Вариант 1.2.

```
int h;
void f(int a, int b)
{
    int x, y;
    x = 3;
    y = 2;
    h = b;
    if (h > y - a) {
        if (x > 0) {
            y = 3;
        }
        h = x;
    }
    if (y > 5) {
        y = 2;
    }
    y = 2;
}

void g(int a, int b)
{
    int x, y;
    x = 9;
    y = 1;
    h = a;
    if (h > x) {
        x = 5;
    }
    h = b - y;
    if (y > 6) {
        y = 2;
        h = a + y;
    }
    while (x > 4) {
        if (h > 0)
            break;
        h = b - a;
        x = 10;
        h = x + y;
    }
}
```

Вариант 1.3.

```
int h;
void f(int a, int b)
{
    int x, y;
    x = 3;
    y = 10;
    h = 1;
    if (h > b - x) {
        if (h > 6) {
            if (x > 7) {
                y = 5;
            }
            x = 8;
            x = 6;
            h = 4;
        }
    }
}

void g(int a, int b)
{
    int x, y;
    x = 4;
    y = 3;
    h = b;
    x = 0;
    if (x < 1) {
        x = 8;
    } else {
        y = 2;
    }
    y = 1;
    while (x > 4) {
        if (h > 0)
            break;
        x = 5;
        if (y > 6) {
            x = 7;
        }
        h = y;
    }
}
```

Вариант 1.4.

```
int h;
void f(int a, int b)
{
    int x, y;
    x = 6;
    y = 5;
    h = 2;
    if (x > 5) {
        if (x < 9) {
            if (h > 6) {
                x = 9;
            }
            x = 7;
        }
    }
    h = 2;
    y = 0;
}

void g(int a, int b)
{
    int x, y;
    x = 1;
    y = 2;
    h = 6;
    if (h < y - a) {
        y = 2;
    }
    y = 3;
    if (h > a) {
        y = 4;
    }
    y = 4;
    while (x < 5) {
        if (h > 0)
            break;
        y = 1;
        if (y < 7) {
            y = 4;
        } else {
            h = x;
        }
    }
}
```

Вариант 1.5.

```
int h;
void f(int a, int b)
{
    int x, y;
    x = 3;
    y = 9;
    h = b;
    if (x > 6) {
        x = 1;
    }
    if (y < 3) {
        if (y < 8) {
            y = 8;
        }
        x = 1;
    }
    h = y;
}

void g(int a, int b)
{
    int x, y;
    x = 2;
    y = 0;
    h = 4;
    if (h > b) {
        x = 4;
    } else {
        x = 3;
    }
    if (x < 3) {
        h = a;
    }
    x = 2;
    while (x > 5) {
        if (h > 0)
            break;
        x = 2;
        if (x > 7) {
            h = a + b;
        }
        h = y;
    }
}
```

Вариант 1.6.

```
int h;
void f(int a, int b)
{
    int x, y;
    x = 2;
    y = 0;
    h = 6;
    if (x > 3) {
        y = 2;
        x = 2;
    } else {
        if (x > 5) {
            if (x < 9) {
                y = 8;
            }
            h = a - y;
        }
    }
}

void g(int a, int b)
{
    int x, y;
    x = 4;
    y = 7;
    h = 3;
    if (x > 6) {
        y = 3;
    } else {
        y = 10;
    }
    x = 2;
    x = 9;
    while (x > 0) {
        if (h > 0)
            break;
        x = 0;
        if (h > 5) {
            y = 0;
        }
        h = 4;
    }
}
```

Вариант 1.7.

```
int h;
void f(int a, int b)
{
    int x, y;
    x = 8;
    y = 9;
    h = 2;
    if (y < 8) {
        if (x < 8) {
            if (h > 5) {
                if (y > 10) {
                    y = 3;
                }
                h = 1;
            }
        }
        x = 2;
    }
    x = 9;
}

void g(int a, int b)
{
    int x, y;
    x = 3;
    y = 6;
    h = b;
    if (x > 6) {
        h = x + y;
        if (h > x) {
            x = 3;
        }
        x = 4;
    }
    x = 4;
    while (x < 4) {
        if (h > 0)
            break;
        h = a;
        if (h > b) {
            y = 4;
        }
        x = 3;
    }
}
```

Вариант 1.8.

```
int h;
void f(int a, int b)
{
    int x, y;
    x = 5;
    y = 3;
    h = 0;
    if (x < 4) {
        h = y;
    } else {
        if (h > b) {
            if (y > 5) {
                x = 10;
            }
            x = 3;
        }
    }
    y = 1;
}

void g(int a, int b)
{
    int x, y;
    x = 2;
    y = 3;
    h = a;
    x = 1;
    if (x < 7) {
        if (x > 7) {
            y = 1;
        }
        h = y;
        x = 1;
    }
    while (x > 7) {
        if (h > 0)
            break;
        y = 7;
        y = 8;
        h = b;
    }
}
```

Вариант 1.9.

```
int h;
void f(int a, int b)
{
    int x, y;
    x = 5;
    y = 9;
    h = 2;
    y = 5;
    if (h > 6) {
        y = 4;
    }
    if (h < a - b) {
        if (y < 3) {
            h = 3;
        }
        y = 7;
    }
}

void g(int a, int b)
{
    int x, y;
    x = 0;
    y = 1;
    h = 2;
    y = 2;
    if (y > 4) {
        y = 2;
    }
    if (x < 5) {
        x = 4;
    }
    x = 3;
    while (x < 1) {
        if (h > 0)
            break;
        if (h < b) {
            x = 7;
        }
        if (y > 6) {
            x = 1;
        } else {
            h = a;
        }
    }
}
```

Вариант 1.10.

```
int h;
void f(int a, int b)
{
    int x, y;
    x = 4;
    y = 2;
    h = 10;
    y = 5;
    if (x > 6) {
        h = y;
    }
    if (x > 2) {
        if (x > 4) {
            x = 2;
        } else {
            y = 6;
        }
    }
}

void g(int a, int b)
{
    int x, y;
    x = 4;
    y = 7;
    h = 5;
    y = 4;
    h = b;
    h = y - x;
    h = a - x;
    while (x > 8) {
        if (h > 0)
            break;
        if (h > y + a) {
            if (x > 6) {
                x = 3;
                x = 4;
            }
            h = y + a;
        }
    }
}
```