Комментарии по выполнению и оформлению контрольной работы №1

Общие замечания для всех задач

- Работу нужно подписать, указать номер группы, а также пронумеровать страницы. На странице с заданием тоже можно писать, а также делать пометки в тексте исходной программы. Если в работе более 2-х страниц, то желательно нумеровать их так, чтобы страницы с соседними номерами можно было проверять, не переворачивая лист.
- Все шаги алгоритмов и вычисления нужно выписывать явно и подробно, если не было указано обратное. Вычисления «методом пристального взгляда», т.е. когда написан готовый ответ без промежуточных выкладок, засчитываться не будут.
- `param_decl x` (объявление формального параметра x функции) считается определением переменной x, и с таких операций начинается первый непустой базовый блок функции (но не Entry, который всегда пустой). Аналогично, операция `return` завершает некоторый базовый блок, из которого исходит дуга в пустой блок Exit. Таких блоков с `return` в конце может быть несколько.
- Для задач достигающих определений и живых переменных, наряду с In_{RD} и Out_{LV} можно также вычислять Out_{RD} и In_{LV}, и использовать систему уравнений, включающие и In, и Out. Если выписывать оба эти множества, вычисления будут более простыми и наглядными, т.к. в одном уравнении будет только передаточная функция, а в другом – только сбор с дуг от предшественников/потомков.

Построение графа потока управления (ГПУ)

- Разметить последовательность команд, выделив базовые блоки в соответствии с определением. Базовые блоки в исходном расположении в последовательности команд должны быть обозначены сверху вниз по порядку буквами латинского алфавита A..Z. ГПУ должен быть изображен на рисунке в виде графа. В этот граф должны быть также добавлены вершины *Entry* и *Exit*.
- Для базовых блоков ГПУ, заканчивающихся условным переходом, та дуга, которая соответствует переходу по условию «истина» должна располагаться справа, по условию «ложь» (*fallthrough дуга*) – слева.
- Выполнить нумерацию базовых блоков по алгоритму, в результате чего блоки получат новую нумерацию B1..Bn. При обходе в первую очередь должна посещаться левая (*fallthrough*) дуга. При этом по шагам должен быть явно выписан порядок посещения дуг/узлов Ai, в какой момент дуга добавляется в остовное дерево, и в какой момент присваивается новый номер вершине Bj.
- На рисунке ГПУ необходимо отметить блоки Bi в новой нумерации, а также выделить дуги, не вошедшие в остовное дерево, пунктиром.

Вычисление множества достигающих определений In_{RD}

- В качестве элементов множеств gen/kill, $In_{RD}\,(Out_{RD})$ нужно указывать просто числа – номера строк (через запятую), в которых определяются переменные (либо в виде d_i, где i соответствует номеру строки с определением). Битовые вектора, как в примере в лекции, использовать не нужно в силу громоздкости такой записи.
- В множество gen для каждого блока должны быть включены только последние определения для каждой переменной в этом блоке (в случае, если блок содержит несколько определений одной переменной). Аналогично и для kill – нет смысла включать в kill определения, которых нет ни в одном gen. Фактически все строки, в которых определяются переменные, позднее переодпределяемые в этом же блоке, можно исключить из рассмотрения еще до начала применения алгоритма, при этом строки перенумеровывать не нужно.
- Множество kill для блока B включает в себя все определения (в пределах рассматриваемой функции) для каждой определяемой в блоке B переменной (но с учетом сказанного в предыдущем пункте). При этом на этапе построения множеств gen/kill не нужно пытаться вычислить, существует ли путь, по которому какое-либо определение может достигать этого блока. В множестве kill также могут «дублироваться» (или нет) определения из gen, это не считается ошибкой, т.к. не влияет на результат.

Вычисление множества живых переменных Out_{LV}

- В качестве элементов множеств обязательно использовать сами переменные, а не их определения. Битовые вектора использовать не нужно. Например: $Out_{LV} = \{ x, y \}$
- Аналогично тому, что сказано для gen и kill на предыдущем слайде, для построения use/def для каждой переменной нужно рассматривать, соответственно, только ее первое использование в блоке и последнее переопределение (остальные просто не будут «видны» за пределами блока, и их можно не рассматривать).
- Для множеств use/def справедливы те же самые рассуждения, которые приведены в последнем пункте предыдущего слайда, соответственно, для множеств gen/kill.

Локальная оптимизация базового блока

- В данной задаче требуется выполнить удаление общих подвыражений, сворачивание констант, удаление мертвого кода и снижение стоимости вычислений. Строить ОАГ, как в примере в лекциях, не требуется.
- Первые три из перечисленных оптимизаций выполняются при построении таблицы значений (ТЗ) и последующем «восстановлении» кода по этой таблице алгоритмами 1.8.1-2. Удаление мертвого кода ***не*** нужно выполнять отдельно до оптимизаций. Снижение стоимости выполняется отдельным проходом в конце, в ходе которого подходящие операции заменяются на более «дешевые».
- Изменения множества LiveNow по мере заполнения ТЗ должны быть показаны по шагам в соответствии с алгоритмом 1.8.1 (можно зачеркивать предыдущее значение, и записывать новое ниже, как в примере в лекциях).
- При заполнении ТЗ нужно надписать в качестве верхнего индекса номер значения (строки таблицы) над переменными в программе, а также (после построения ТЗ) надписать номера значений переменным в множестве Output.
- При «восстановлении» кода из ТЗ необходимо, начав с номеров значений в Output, последовательно отмечать (например, «галочкой») строки в ТЗ, содержащие «живые» значения, а также строки в базовом блоке, в которых определяется соответствующая значению переменная. Затем нужно перейти к номерам строк ТЗ, записанным в аргументах операции в «живой» строке, и т.д. В конце нужно выписать заново базовый блок, содержащий только помеченные инструкции.