**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**
**Федеральное государственное автономное образовательное учреждение высшего образования**
**«Севастопольский государственный университет»**
**Институт информационных технологий и управления в технических системах**

# МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению контрольной работы № 1

## «Прямые методы трансляции»

по дисциплине “Системное программное обеспечение”

для студентов по направлению подготовки
09.03.01 «Информатика и вычислительная техника»
Направленность (профиль)
09.03.01.01 «ЭВМ, системы и сети»
заочной формы обучения

Севастополь
2016

УДК 519.6

Методические указания к выполнению контрольной работы №1 по дисциплине “Системное программное обеспечение” для студентов направления подготовки 09.03.01. «Информатика и вычислительная техника» заочной формы обучения/Сост. доцент кафедры ИТиКС С.Н. Фисун, доцент кафедры ИТиКС Ю.Г. Шаталова– Севастополь: Изд-во СГУ, 2016. – 16 с.

Целью методических указаний является оказание помощи студентам в освоении теоретического материала и выполнении контрольной работы № 1 по дисциплине «Системное программное обеспечение». Содержат цели, задачи и учебную нагрузку по дисциплине, краткое изложение основных теоретических положений, задания на контрольную работу, порядок ее выполнения, список рекомендованной литературы.

Методические указания рассмотрены и утверждены на заседании кафедры ИТиКС (протокол № 3 от 6 ноября 2015 г.).

Допущено учебно-методической комиссией института ИТ и УТС СевГУ в качестве методических указаний.

Рецензент:
Карапетьян В.А., доцент кафедры информатики и управления в технических системах, кандидат технических наук

СОДЕРЖАНИЕ



# Введение

В контрольной работе№1 студент должен выполнить пять заданий.

Вариант задания для выполнения контрольной работы выбирается по последней цифре зачетной книжки.

Теоретический материал, необходимый для выполнения контрольной работы, содержится в разделе 3 методических указаний по изучению теоретических основ дисциплины (которые размещены в той же папке, где и методические указания к выполнению данной контрольной работы).

Работа должна быть выполнена аккуратно, желательно в виде компьютерного набора. Однако не запрещается выполнять рисунки и схемы вручную, но обязательно с использованием линейки.

Разрешается выполнять расчеты как вручную, так и с использованием любых средств автоматизации вычислений (могут быть использованы любые пакеты прикладных программ). Соответствующие листинги должны быть внедрены в текст контрольной работы как «объект» с дополнительными пояснениями.

На титульном листе обязательно должны присутствовать номер зачетной книжки и номер выполняемого варианта. Образец титульного листа приведен в приложении 1.

Работа должна включать следующие разделы:

- Содержание.
- Задание №1 Построение древовидной и ХЕШ-таблицы для заданной последовательности ключей.
- Задание №2. Сортировка элементов массива с использованием алгоритмов Хоара и Шелла .
- Задание №3. Разработка и отладка программы построения таблицы или упорядочения элементов массива
- Задание №4. Построить ПОЛИЗ для заданного выражения.
- Задание №5 Разработать и отладить программу построения ПОЛИЗ.

Для каждого из заданий необходимо представить постановку задачи, результаты решения задачи и анализ полученных результатов.

Работы, выполненные небрежно или не для своего варианта задания будут возвращены без проверки!!!

# 1 ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ

В результате изучения дисциплины студенты должны знать методы проектирования СПО, методы лексического и синтаксического анализа, генерации и оптимизации кода.

Преподавание дисциплины преследует следующие цели:

- ознакомление с составом и структурой СПО, принципами его организации;
- изучение различных методов проектирования СПО, методов лексического и синтаксического анализа, методов генерации и оптимизации кода.

В результате изучения дисциплины студент должен знать:

- состав и структуру СПО;
- прямые и синтаксические методы трансляции;
- методы лексического и синтаксического анализа;
- методы генерации и оптимизации кода;
- методы разработки, отладки и тестирования СПО

В результате изучения дисциплины студент должен овладеть навыками анализа грамматик формальных языков, разработки блоков трансляторов.

Дисциплина связана с предшествующими ей дисциплинами «Высшая математика», «Программирование», «Системное программирование», «Алгоритмизация и формальные преобразования»и является базовой для дисциплин «Операционные системы» , «Теория проектирования ЦВМ», «Моделирование».

# 2 УЧЕБНАЯ НАГРУЗКА ПО ДИСЦИПЛИНЕ

В соответствии с учебным планом дисциплина изучается в пятом, шестом и седьмом семестрах в соответствии с таблица 2.1

| Семестр | Лекции | Лаб.зан | Контр.раб | Курс. проект | Зачет | Экзамен |
|---|---|---|---|---|---|---|
| 5 | 4 | | | | | |
| 6 | 4 | 4 | КР1 | | | Экзамен |
| 7 | 12 | 10 | КР2 | КП | | |

На протяжении 6-го и 7-го семестров проводятся консультации каждую нечетную субботу в соответствии с расписанием.

**В пятом семестре** проводятся две установочные лекции и выдаются задания на контрольную работу №1«Прямые методы трансляции». Выполнение контрольной работы №1 предполагает решение задач сортировки и поиска информации и построения польской инверсной записи (ПОЛИЗ). Номер варианта в методических указаниях выбирается по последней цифре зачетки. Варианты заданий на контрольную работу №1 приведены в разделе 4 настоящих методических указаний.

**В шестом семестре** проводятся два двухчасовых лабораторных занятия:

–Лабораторная работа №1 «Методы сортировки и поиска информации в таблицах транслятора»;

–Лабораторная работа №2 «Трансляция выражений».

Формулировка задания на лабораторные работы, порядок выполнения и форма отчетности приведены в [4],[10],[15].

**Необходимым условием** для сдачи экзамена в шестом семестре является качественно выполненные и защищенные контрольная работа №1 и лабораторные работы №1 - №2.

**В седьмом семестре** проводятся два четырехчасовых лабораторных занятия:

–Лабораторная работа № 3 «Программирование сканера»;

–Лабораторная работа № 4 «Синтаксический анализ методом рекурсивного спуска».

Формулировка задания на лабораторные работы, порядок выполнения и форма отчетности приведены в [4],[10],[11].

**Необходимым условием** для защиты курсового проекта является качественно выполненные: курсовой проект на тему «Разработка отладка и тестирование блока транслятора для подмножества учебного языка высокого уровня» [12], контрольная работа №2[11] и лабораторные работы №3 - №4 [10].

# 3 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

## 3.1 Общая схема компиляции

Транслятор—это программа, которая переводит программу на исходном языке в эквивалентную ей программу на выходном языке.

Компилятор – это транслятор, который осуществляет перевод исходной программы в эквивалентную ей результирующую программу на языке машинных кодов или на языке ассемблера.

Интерпретатор—это программа, которая воспринимает исходную программу на входном языке и выполняет ее.

Задача компилятора состоит в преобразовании ориентированного на пользователя программного обеспечения в машинно-ориентированное представление.

Задача компиляции рассматривается на двух этапах, каждый из которых состоит из фаз:

1. Этап анализа
1.1. Лексический анализ.
1.2. Синтаксический анализ.
1.3. Семантический анализ.
2. Этап синтеза.
2.1. Генерация машинно-независимого кода.
2.2. Оптимизация машинно-независимого кода.

2.3. Распределение памяти.
2.4. Генерация машинного кода.
2.5. Оптимизация машинного кода.

Общая схема компиляции представлена на рисунке 3.1.

## 3.2 Задачи фаз компиляции

Лексический анализ. На этой фазе формируются лексемы языка.

Лексема—это структурная единица языка, которая состоит из элементарных символов языка и не содержит в своем составе других структурных единиц языка. Для языков программирования лексемами являются идентификаторы, константы, ключевые слова, знаки операций и т. д.

Задачей фазы является переход от символов входного языка к лексемам, а также проверка корректности входной строки. На вход лексического анализатора подается строка, состоящая из символов входного языка, происходит преобразование введенных символов по предусмотренным языком шаблонам в лексемы, которые кодируются и передаются на последующие фазы, например, ключевые слова if, while, do, then, а также знаки присвоения **:=,** равенства = = и т.д. рассматриваются как единые сущности (лексемы) и представляются кодами, например, i, w, d, t, p, r соответственно. На дальнейших фазах компиляции работа ведется именно с лексемами, а не с отдельными символами. Лексический анализатор не работает с контекстом, для него не важен порядок символов, он анализирует только принадлежность введенных символов языку, а также соответствие введенных конструкций шаблонам языка. Наряду с этим происходит удаление пробелов, комментариев и других символов, не имеющих смысловой нагрузки.

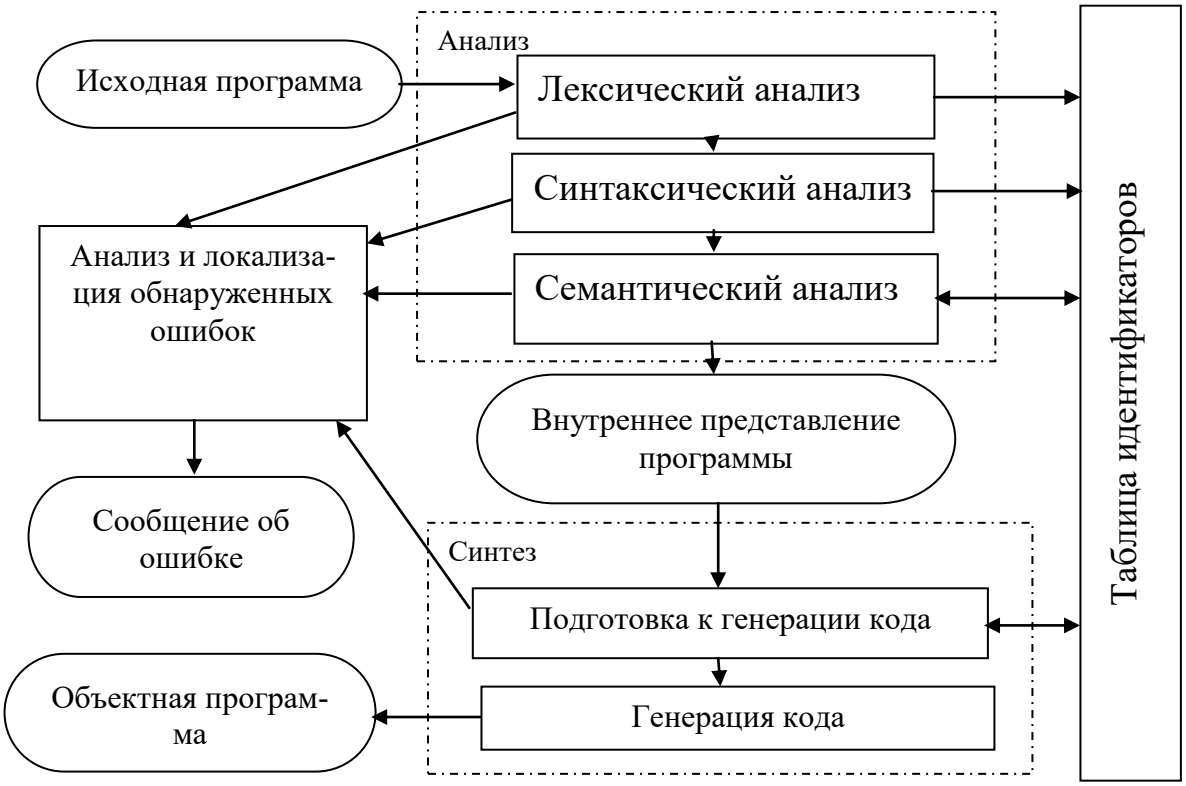


Рисунок 3.1 – Общая схема работы компилятора

На фазе лексического анализа начинается формирование таблицы идентификаторов.

Синтаксический анализ. Работает с контекстом. Определяет общую структуру конструкций языка. Например, может проверить соответствие скобок, корректность структур if –then-else, while- do, и т.д. Результатом синтаксического анализа является представление программы в виде синтаксического дерева, листья которого соответствуют символам входной строки. На этом этапе продолжает формироваться таблица идентификаторов.

Семантический анализ. Выполняет проверку соответствия типов данных в исходной программе, отлавливает бесконечные рекурсии, деление на ноль и другие недопустимые действия.

На этапе анализа ведущей фазой является синтаксический анализ. Обычно разработанная программа – это программа синтаксического анализатора, по необходимости вызывающая модули лексического и семантического анализа.

На этапе синтеза не все фазы являются обязательными. Например, можно не строить промежуточный (машинно-независимый) код, можно обойтись без фазы оптимизации на уровне машинно-независимого кода, на уровне машинно-зависимого кода или на обоих. Но есть ряд причин, по которым эти фазы присутствуют в процессе компиляции. Рассмотрим эти причины.

Генерация машинно-независимого (промежуточного) кода. Включение этой фазы в процесс компиляции позволяет:

- обеспечить переносимость компилятора на другие машины, поскольку позволяет обособить компилятор от целевого языка и конкретной машины;
- облегчить процесс построения целевого кода, т.к. строить машинно-зависимый код гораздо удобнее по промежуточному коду;
- обеспечить дополнительный уровень оптимизации синтезируемого кода.

В качестве промежуточного представления могут использоваться следующие средства: синтаксическое дерево разбора, полученное в результате синтаксического анализа; ПОЛИЗ (польская инверсная запись), представление кода в виде триад и т.д.

Оптимизация. Потребность в оптимизации может быть различной. Если требуется очень эффективный код, то компилятор должен обеспечить значительную оптимизацию. В то же время, во многих средах скорость работы ПО не является критическим параметром, следовательно, достаточно лишь незначительной оптимизации. Поскольку многие типы оптимизации трудоемки и требуют значительных затрат, то выбор уровня оптимизации является серьезной задачей. В некоторых компиляторах пользователь сам может выбрать уровень оптимизации.

Распределение памяти. На этой фазе каждая константа и переменная, используемые в программе получают зарезервированное место в памяти для хранения своего значения. Данная область памяти может иметь один из следующих типов:

1). Статическая память, если время жизни переменной равно времени жизни программы. Не может быть освобождена до завершения выполнения программы.

2). Динамическая память, если время жизни переменной равно времени жизни определенного блока, функции или процедуры. Может быть освобождена после выполнения данного фрагмента, до завершения выполнения программы.

3). Глобальная матрица, если время жизни переменной в момент компиляции неизвестно, а память должна освобождаться и выделяться в процессе выполнения по необходимости.

От эффективности распределение памяти  во многом зависит эффективность работы компилятора.

## 3.3 Хранение и поиск данных в таблицах[4,7]

Главной характеристикой любого элемента исходной программы является его имя. Задачей любого транслятора является необходимость хранения и доступа к информации, связанной с каждым элементом исходной программы. Для решения этой задачи транслятор организует специальные хранилища данных, называемые таблицами идентификаторов или таблицами символов. Количество элементов в таблицах велико и поэтому время поиска информации в них существенно влияет на общее время трансляции. Поэтому таблицы идентификаторов должны быть организованы таким образом, чтобы транслятор выполнял операции поиска информации в таблицах с максимальной эффективностью. Для этого используются следующие способы организации таблиц [4,7]:

- бинарные деревья;
- хеш-адресация с открытым(случайным) перемешиванием;
- хеш-адресация с цепочками переполнения;
- упорядоченные списки.

Древовидные таблицы являются улучшенным вариантом неупорядоченных таблиц. Новые записи в древовидную таблицу по-прежнему поступают неупорядоченно. Но на этот раз место, занимаемое этими записями среди других записей в таблице, определяется значением ключевого поля. Принцип здесь такой же, как и в сортировке с помощью дерева. В общем случае каждая запись в древовидной таблице сопровождается двумя указателями:

-один указатель хранит адрес записи с меньшим значением ключа (адрес узла-предка);

-второй указатель хранит адрес записи с большим значением ключа (адрес узла-сына).

Процесс добавления нового элемента в древовидную таблицу производится следующим образом. Значение ключа нового элемента сравнивается со значением ключа первого элемента дерева. По результатам сравнения ключей новый элемент «поступает» в левое или правое поддерево первого элемента дерева. Далее процесс сравнения и поступления в левое или правое поддерево очередного узла продолжается аналогичным образом и до тех пор, пока не будет найден пустой элемент в нужном направлении просмотра.

Новая запись помещается в данный пустой элемент, при этом настраивается указатель с адресом узла-предка. Преимущества древовидной структуры таблицы проявляются на этапе поиска нужного элемента. Поиск прекращается либо при совпадении ключевого поля, либо при обнаружении пустого указателя.

Таблицы с вычисляемыми входами (хеш-таблицы) подробно рассмотрены в [7] на страницах 47-53.

## 3.4 Упорядочивание данных в таблицах

Упорядочивание данных в таблицах производится с использованием одного из алгоритмов сортировки. В трансляторах наиболее широко применяются алгоритмы Шелла и Хоара [8,9]. Алгоритм Шелла – сортировка с убывающим шагом. При сортировке Шелла сначала сравниваются и сортируются между собой значения, отстоящие один от другого на некотором расстоянии $d$. После этого процедура повторяется для некоторых меньших значений $d$, пока $d$ не станет отрицательным. Структурная диаграмма (диаграмма Несси-Шнейдермана) для алгоритма Шелла представлена на рис. 3.2. Алгоритм Хоара подробно рассмотрен в [8] на страницах 114 – 124.

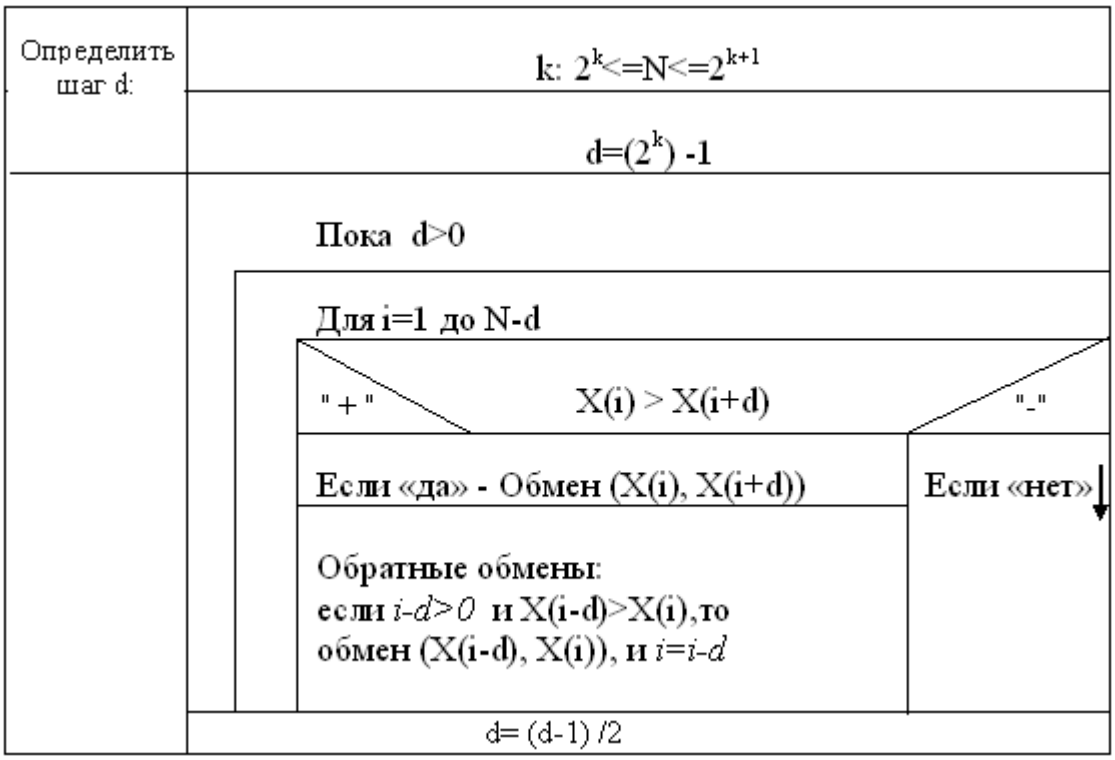


Рисунок 3.2 – Структурная диаграмма для алгоритма Шелла

## 3.5 Трансляция выражений

Польская инверсная запись (ПОЛИЗ) названа в честь польского математика Яна Лукашевича, который впервые использовал эту форму для представления выражений в математической логике. Она обладает следующими свойствами:

- операнды располагаются в том же порядке, что в исходном выражении;
- знаки операций располагаются в том порядке, в котором вычисляется выражение;
- знак каждой операции записывается после соответствующих операндов.

Для построения ПОЛИЗ можно использовать приведенные выше свойства. На практике широко используется метод, который предложен в 1960 году голландским ученым Е.В.Дейкстрой [7]. Этот метод основан на использовании стека с приоритетами, позволяющего изменить порядок следования знаков операций в выраже-

нии так, что получается ПОЛИЗ. Простейший вариант этого метода применим только к простым арифметическим и логическим выражениям, содержащим простые переменные, знаки арифметических и логических операций, знаки операций отношения и круглые скобки. Каждому ограничителю, входящему в выражение, присваивается приоритет. Для знаков операции приоритеты возрастают в порядке, обратном старшинству операций. Скобки имеют низший приоритет. Арифметическое или логическое выражение рассматривается как входная строка символов. Входная строка рассматривается слева направо. Операнды переписываются в выходную строку, а знаки операций помещаются вначале в стек операций. Если приоритет входного знака равен нулю или больше приоритета знака, находящегося в вершине стека, то новый знак добавляется к вершине стека. В противном случае из стека "выталкивается" и переписывается в выходную строку знак, находящийся в вершине, а также следующие за ним знаки с приоритетами большими или равными приоритету входного знака. После этого входной знак добавляется к вершине стека.

Особенности имеет лишь обработка скобок. Открывающаяся круглая скобка, имеющая "особый" приоритет нуль, просто записывается в вершину стека и не выталкивает ни одного знака. В то же время ее не может вытолкнуть ни один знак, кроме закрывающей скобки.

Закрывающая скобка имеет приоритет 1, не превосходящий приоритет любой операции. Поэтому появление закрывающей скобки вызывает выталкивание всех знаков до ближайшей открывающей скобки включительно. В стек закрывающая скобка не записывается. Открывающая и закрывающая скобки как бы взаимно уничтожаются и в выходную строку не переносятся.

После просмотра всех символов входной строки происходит выталкивание всех оставшихся в стеке символов и дописывание их к выходной строке.

В контрольной работе в вариантах заданий в выражениях используются переменные с индексами и указатели функций.

Для перевода выражений, содержащих переменные с индексами, также применим стек с приоритетами. Индексные скобки “[“ и “]” в некотором смысле играют ту же роль, что и круглые скобки. Действие этих скобок на стек, как и действие круглых скобок, состоит, прежде всего, в следующем - открывающая индексная скобка всегда записывается в вершину стека, а закрывающая индексная скобка выталкивает из стека все знаки до ближайшей открывающей индексной скобки включительно.

Однако в отличие от круглых скобок одновременно с записью в стек открывающей индексной скобки туда записывается также начальное (минимальное) значение счетчика операндов операции адрес элемента массива (АЭМ), равное 2. Кроме того, после выталкивания из стека знаков закрывающей индексной скобкой и переписи их в выходную строку туда же переписывается значение счетчика операндов, а затем и сама закрывающая скобка.

Запятая, разделяющая индексные выражения, играет одновременно роль закрывающей скобки для предыдущего индексного выражения и роль открывающей скобки - для последующего. Поэтому запятой можно назначить приоритет 1 как закрывающей скобке, дополнив алгоритм условием, что запятая выталкивает из стека

все знаки операции до ближайшей открывающей индексной скобки исключительно. Сама запятая, как и любая закрывающая скобка, в стек не записывается. Появление запятой равносильно появлению еще одного индекса, поэтому каждая запятая добавляет в счетчик операндов операции АЭМ единицу.

Алгоритм перевода в ПОЛИЗ функции, имеющий не менее одного параметра, тот же, что для переменной с индексом. Различие состоит лишь в том, что в момент прихода закрывающей круглой скобки в выходную строку записывается символ Ф. Чтобы отличить открывающую круглую скобку в начале списка фактических параметров от открывающей круглой скобки в начале выражения, можно использовать переменную состояния F (признак функции). Эта переменная обычно имеет значение нуль. В момент появления идентификатора функции она принимает значение 1, а после занесения в стек круглой скобки и начального значения счетчика операндов, равного 2, вновь принимает значение 0. Закрывающая скобка, встретив в стеке открывающую круглую, записанную вместе со значением счетчика операндов, занесет это значение в выходную строку, запишет туда знак Ф и уничтожит в стеке круглую скобку и значение счетчика операндов.

Примеры перевода арифметических выражений рассмотрены в [7] на страницах 130 – 144.

# 4 ЗАДАНИЯ КОНТРОЛЬНОЙ РАБОТЫ №1

Контрольная работа №1 содержит два задания, которые имеют непосредственное отношение к повышению эффективности процесса поиска информации в таблицах. В задании №1 необходимо построить древовидные и хеш-таблицы для заданной последовательности ключей. В задании №2 рассматривается сортировка списков с использованием эффективных алгоритмов Шелла и Хоара. Трансляция выражений – одна из основных задач транслятора. Она рассматривается в задании №4. В настоящее время классическим является метод трансляции выражений, основанный на использовании польской инверсной записи (ПОЛИЗ). Этот метод хорошо иллюстрирует идею использования прямых методов трансляции в современных системах программирования. Суть этих методов – использование некоторой руководящей общей идеи, применение которой позволяет для каждой языковой конструкции разработать индивидуальный алгоритм трансляции. Задания с номерами 3 и 5 посвящены разработке программ на языках Java (C) для выполнения на ПЭВМ заданий 1,2,4.

**4.1.** Построить древовидную и ХЕШ-таблицы для заданной последовательности ключей. Размер ХЕШ-таблицы — N+6. (N - длина заданной последовательности ключей).

ХЕШ-функция определяется как: (остаток целочисленного деления ключа на N)+1. Способ устранения коллизий выбрать в соответствии с вариантом задания:

1 - открытое перемешивание;

2 - перемешивание с цепочками переполнения. Варианты заданий представлены в табл.1.

**4.2.** Упорядочить элементы массива с использованием алгоритмов Хоара и Шелла .В качестве массива рассмотреть последовательность ключей в соответствии с вариантом задания в табл.4.1.

**Таблица 4.1**

Варианты заданий к задачам 4.1, 4.2.

| № варианта | Последовательность ключей | Способ устранения коллизий в ХЕШ-таблице |
|---|---|---|
| 1 | 2 | 3 |
| 0 | 13,3,17,4,5,2,1,8,9,6,12 | 1 |
| 1 | 12,4,16,5,7,13,2,14,11,9,8 | 1 |
| 2 | 11,5,15,1,3,2,9,8,7,6,5 | 1 |
| 3 | 10,6,14,13,12,11,9,1,2,4,3 | 1 |
| 4 | 9,7,13,10,8,4,3,2,1,14,15 | 1 |
| 5 | 8,9,12,13,14,2,1,5,4,3,6 | 2 |
| 6 | 7,8,11,1,16,15,14,13,12,5,4 | 2 |
| 7 | 6,10,13,17,21,1,3,5,11,9,7 | 2 |
| 8 | 5,11,12,10,9,8,7,6,4,3,2 | 2 |
| 9 | 4,12,11,1,2,7,8,6,5,22,3 | 2 |

**4.3.** Написать и отладить на ПЭВМ типа IBM PC программу на языке JAVA (СИ) для решения задачи построения таблицы или упорядочения элементов массива. Варианты заданий представлены в табл.2.

**4.4.** Построить ПОЛИЗ для заданного выражения. Варианты заданий представлены в табл.3.

**4.5**. Разработать и отладить программу построения ПОЛИЗ в соответствии с вариантом задания в табл.4.

**Таблица 4.2**

Варианты заданий к задаче 4.3

| № вариан-та | Наименование задачи | Тип таблицы – алгоритм сортировки | Язык програми-рования |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 0 | Построение таблицы | Древовидная | JAVA |
| 1 | Построение таблицы | Древовидная | СИ |
| 2 | Построение таблицы | ХЕШ-табл (откр. перемеш) | JAVA |
| 3 | Построение таблицы | ХЕШ-табл(откр. перемеш) | СИ |
| 4 | Упорядочение элементов | Алгоритм Шелла | JAVA |
| 5 | Упорядочение элементов | Алгоритм Шелла | СИ |
| 6 | Упорядочение элементов | Алгоритм Хоара | JAVA |
| 7 | Упорядочение элементов | Алгоритм Хоара | СИ |
| 8 | Построение таблицы | ХЕШ-табл (цеп. переп) | JAVA |
| 9 | Построение таблицы | ХЕШ-табл (цеп. переп) | СИ |

**Таблица 4.3**

Варианты заданий к задачам 4.4

| № варианта | Выражение |
|---|---|
| 0 | X[I,J]*SIN(A+(A-B[I])*EXP(C+D/B[I])) |
| 1 | A[I]*EXP(C+D)*(SQR(C*C+D*D)+C-A[I]^3) |
| 2 | B[I,K,R]+EXP(A[I]^2+B[I]^2+SQR(C-D)) |
| 3 | R*(1-R*K[I+2,J*J-3]+COS(X[I]+A[I]^2) |
| 4 | A+B[R,S]*(SIN(X[I]+A)^2+COS(X[I+A]^2) |
| 5 | R-A*(B-(C/(A-B))*COS(X[I,J]+X[I+1,J+1]) |
| 6 | SQR(A+B*(X[I*K+J*K]-1/SIN(A+Y[I,J]) |
| 7 | R*A*EXP(X[I]^2+X[I+1]^3+R*B[I,J,K]) |
| 8 | K*LOG(X[I,J]^2+Y^2)*X[I+1,J*J]*SIN(X) |
| 9 | B[I]-C*(A[I]+B[I]*X[I,J]*(COS(A[I]+B[I])) |

**Таблица 4.4**

Варианты заданий к задаче 4.5

| № варианта | Тип выражения | Наименование задачи | Язык программирования |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 0 | Простое арифметическое JAVA | Построение ПОЛИЗ | JAVA |
| 1 | Простое арифметическое JAVA | Построение ПОЛИЗ | СИ |
| 2 | Простое логическое | Построение ПОЛИЗ | JAVA |
| 3 | Простое логическое | Построение ПОЛИЗ | СИ |
| 4 | Переменная с индексами | Построение ПОЛИЗ | JAVA |
| 5 | Переменная с индексами | Построение ПОЛИЗ | СИ |
| 6 | Указатель функции | Построение ПОЛИЗ | JAVA |
| 7 | Указатель функции | Построение ПОЛИЗ | СИ |
| 8 | Простое арифметическое СИ | Построение ПОЛИЗ | JAVA |
| 9 | Простое арифметическое СИ | Построение ПОЛИЗ | СИ |

# 5 СПИСОК ВОПРОСОВ ДЛЯ ПОДГОТОВКИ К ЭКЗАМЕНУ

5.1. Основные понятия и функции системного ПО ЭВМ (СПО ЭВМ).
5.2. Задачи, возникающие при разработке СПО ЭВМ.
5.3. Организация, хранение и поиск данных.
5.4. Структуры данных.
5.5. Структуры хранения.
5.6. Отображение структур данных в структуры хранения.
5.7. Хранение и поиск данных в таблицах. Методы хеширования.
5.8. Сортировка. Классификация методов и сравнительный анализ. Методы внутренней сортировки.
5.9. Трансляторы. Виды трансляторов. Общая схема трансляции.
5.10. Лексический анализ. Задачи и функции. Перекодирование входной программы и формирование лексем с использованием прямых методов.
5.11. Трансляция выражений и операторов в машинно-независимое промежуточное представление (ПОЛИЗ, триады).
5.12. Машинно-независимая оптимизация объектной программы.
5.13. Генерация и оптимизация объектного кода.
5.14. Блоки и трансляция описаний.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Компиляторы: принципы, технологии и инструментарий. / А. Ахо, М. Лам, Р. Сети, Дж. Д. Ульман. - М.: Вильямс, 2008. – 1184 с.
2. Свердлов С.З. Языки программирования и методы трансляции: Учебное пособие. / С.З. Свердлов - СПб.: Питер, 2007.— 638 с.
3. Системное программное обеспечение: Учебник для вузов. / А.Ю Молчанов. – СПб.: Питер, 2009. – 396 с.
4. Системное программное обеспечение. Лабораторный практикум./ А.Ю. Молчанов. - СПб.: Питер, 2005. – 284 с.
5. Опалева Э.А. Языки программирования и методы трансляции. / Э.А. Опалева, В.П. Самойленко— СПб.: БХВ-Петербург, 2005. — 480 с.
6. Карпов Ю.Г. Теория и технология программирования. / Ю.Г. Карпов.- СПб.: БХВ-Петербург, 2005. — 272 с..
7. Лебедев В.Н. Введение в системы программирования./ В.Н. Лебедев М.: Символ-Плюс, 2015.— 312 с.
8. Вирт Н. Алгоритмы и структуры данных. / Н. Вирт. - СПб.: Невский Диалект, 2008. - 352 с.
9. Кнут Д. Искусство программирования для ЭВМ. Том 3. Сортировка и поиск. / Д. Кнут - М.: Вильямс, 2012. –824 с.
10. Методические указания к выполнению лабораторных работ по дисциплине «Системное программное обеспечение» для студентов очной и заочной форм обучения по направлению подготовки 09.03.01 «Информатика и вычислитель-

ная техника». / Сост. доцент кафедры КиВТ Фисун С.Н. – Севастополь: Изд-во СевНТУ, 2014. -64с.

11. Методические указания к выполнению контрольной работы № 2 по дисциплине «Системное программное обеспечение» для студентов заочной формы обучения по направлению 09.03.01 «Информатика и вычислительная техника» / Сост. доцент кафедры КиВТ Фисун С.Н., доцент кафедры КиВТ Шаталова Ю.Г. - Севастополь: Изд-во СевНТУ, 2014. – 14с.
12. Методические указания к выполнению курсового проекта по дисциплине «Системное программное обеспечение» для студентов очной и заочной формы обучения по направлению 09.03.01 «Информатика и вычислительная техника» / Сост. доцент кафедры КиВТ Фисун С.Н., доцент Сергеев Г.Г. - Севастополь: Изд-во СевНТУ, 2014. – 74с.

# ПРИЛОЖЕНИЕ

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**
**Федеральное государственное автономное образовательное учреждение высшего образования**
**«Севастопольский государственный университет»**
**Институт информационных технологий и управления в технических системах**

Кафедра информационных
технологий и компьютерных систем

КОНТРОЛЬНАЯ РАБОТА №1
на тему: «Прямые методы трансляции»
по дисциплине «Системное программное обеспечение»
№ зачётной книжки ***xxxxxxxx***
№ варианта ***xx***

Выполнил: студент(ка) гр. ИВТб-***XX***з
***xxxxxxxxx x.x.***
Проверил(а):

Севастополь
2016

Заказ № _____ от «  » _____ 2016 Тираж _____ экз
Изд-во СГУ