Частное учреждение образования

«Колледж бизнеса и права»

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Ведущий методист колледжа  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.В. Паскал  « \_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 года |
| Специальность 2-40 01 01 «Программное обеспечение информационных технологий» | Учебная дисциплина «Основы алгоритмизации и программирование» |

**Лабораторная работа № 37**

**Инструкционно-технологическая карта**

Тема: Разработка алгоритмов и программ с использованием комбинаторных алгоритмов, алгоритмов на графах, динамических структур данных.

Цель: Научиться создавать алгоритмы и программы с использованием комбинаторных алгоритмов, алгоритмов на графах, динамических структур данных.

Время выполнения: 2 часа.

1. **Порядок выполнения работы**
2. Изучить теоретические сведения к лабораторной работе.
3. Разработать на языке С++ программу вывода на экран решения задачи в соответствии с вариантом индивидуального задания, указанным преподавателем.
4. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.
5. Ответить на контрольные вопросы.
6. **Теоретические сведения**

**Построение обратной польской записи с помощью дерева и стека**

Сложные вычислительные задачи обычно требуют больших объемов вычислений, поэтому к разработчикам языков программирования было предъявлено требование: максимально приблизить форму записи математических выражений в коде программы к естественному языку математики. Одну из первых областей системного программирования составили исследования способов трансляции математических выражений. Наибольшее распространение получил метод трансляции при помощи **обратной польской записи** (**ОПН** – обратная польская нотация), которую предложил польский математик Я. Лукашевич. Рассмотрим алгоритмы получения обратной польской записи с использованием динамических структур данных в виде дерева и стека.

**Алгоритм, использующий дерево**

Данный алгоритм основан на представлении математического выражения в виде дерева и использовании способа обхода дерева снизу вверх (сначала обходим левое поддерево, потом – правое поддерево и последним посещаем корень). Рассмотрим его на примере арифметического выражения (A+B)\*(C+D)–E.

Представим это выражение в виде дерева, в котором узлам соответствуют операции, а листьям – операнды. Построение начинается с корня, в качестве которого выбирается последняя выполняемая операция. Левой ветви соответствует левый операнд операции, а правой ветви – правый. Дерево выражения имеет вид:



Совершим обход дерева, под которым понимается формирование строки из символов узлов и ветвей дерева. Обход будем совершать от самой левой ветви вправо и узел переписывать в выходную строку только после рассмотрения всех его ветвей. Обход совершаем строго по уровням:

1) уровень 2: АВ+CD+

2) поднялись на уровень 1: \*Е

3) и, наконец, корень: –

В результате такого обхода получили обратную польскую запись: AB+CD+\*E–

**Алгоритм, использующий стек**

Получение обратной польской записи с использованием стека может осуществляться на основе алгоритма, предложенного Дейкстрой, который ввел понятие **стекового приоритета операций**:

|  |  |
| --- | --- |
| **Операция** | **Приоритет** |
| ( | 1 |
| + – | 2 |
| \* / | 3 |

Суть алгоритма в следующем. Просматривается исходная строка символов S слева направо, операнды переписываются в выходную строку В, а знаки операций заносятся в стек, который первоначально пуст, на основе следующих правил:

1) если в строке S встретился операнд, то помещаем его в строку В;

2) если в S встретилась открывающая скобка, то помещаем ее в стек;

3) если в S встретилась закрывающая скобка, то выталкиваем из стека в строку В все операции до открывающей скобки, саму отрывающую скобку также извлекаем из стека; обе скобки (открывающая и закрывающая) игнорируются;

4) если в S встретилась операция Х, то выталкиваем из стека все операции, приоритет которых не ниже Х, после чего операцию Х записываем в стек;

5) при достижении конца строки S, если стек не пуст, переписываем его элементы в выходную строку В.

Обратная польская запись обладает рядом замечательных свойств, которые превращают ее в идеальный промежуточный язык при трансляции кода. Во-первых, вычисление выражения, записанного в обратной польской записи, может проводиться путем однократного просмотра, что является весьма удобным при генерации объектного кода программ. Вычисление полученного выражения может быть проведено следующим образом:

|  |  |  |
| --- | --- | --- |
| **Шаг** | **Анализируемая строка** | **Действие** |
| 1 | AB+CD+\*E– | R1=A+B |
| 2 | R1CD+\*E– | R2=C+D |
| 3 | R1 R2\*E– | R1=R1\*R2 |
| 4 | R1 E– | R1=R1–E |
| 5 | R1 |  |

Здесь R1 и R2 – вспомогательные переменные.

**Пример реализации польской записи, используя стек**

Пусть задано выражение: a+b\*c+(d\*e+f)\*g. Необходимо записать это выражение в постфиксной форме. Правильным ответом будет выражение: abc\*+de\*f+g\*+. Решаем эту задачу, используя стек.

Пусть исходная информация хранится в строке S=”a+b\*c+(d\*e+f)\*g”. Результат будем получать в строке В.

Начинаем последовательно просматривать символы исходной строки, причем стек пуст и В – пустая строка.

Букву «a» помещается в строку В, а операцию «+» помещаем в стек. Букву «b» помещаем в строку В. На этот момент стек и строка В выглядят следующим образом:

|  |  |
| --- | --- |
|  | В = ”ab” |
| + |  |

Операцию «\*» помещаем в стек, т.к. элемент в вершине стека имеет более низкий приоритет. Букву «с» помещаем в строку В, после чего имеем:

|  |  |
| --- | --- |
|  | В = ”abс” |
| \* |  |
| + |  |

Следующий символ строки S «+». Анализируем стек и видим, что элемент в вершине стека «\*» и следующий за ним «+» имеют приоритеты не ниже текущего, следовательно, обе операции извлекаем из стека и помещаем в строку В, а текущий элемент помещаем в стек.

|  |  |
| --- | --- |
|  | В = ”abс\*+” |
| + |  |

Далее в строке S следует символ «(», его помещаем в стек, а букву «d» помещаем в строку В, в результате получается:

|  |  |
| --- | --- |
|  | В = ”abс\*+d” |
| ( |  |
| + |  |

Следующий в строке S символ «\*». Так как открывающую скобку нельзя извлечь из стека до тех пор, пока не встретилась закрывающая, то «\*» помещаем в стек. Букву «e» помещаем в строку В:

|  |  |
| --- | --- |
|  | В = ”abс\*+de” |
| \* |  |
| ( |  |
| + |  |

Следующий прочитанный символ «+», и т.к. элемент стека «\*» имеет более высокий приоритет, то извлекаем его из стека и помещаем в строку В, а текущий символ «+» помещаем в стек. Символ «f» помещаем в строку В:

|  |  |
| --- | --- |
|  | В = ”abс\*+de\*f” |
| + |  |
| ( |  |
| + |  |

Далее в строке S идет закрывающая скобка, все элементы стека до символа «)» помещаем в строку В (это элемент «+»), а сам символ «(» извлекаем из стека. Обе скобки игнорируются:

|  |  |
| --- | --- |
|  | В = ”abс\*+de\*f+” |
| + |  |

Операцию «\*» помещаем в стек, а букву «g» – в строку В:

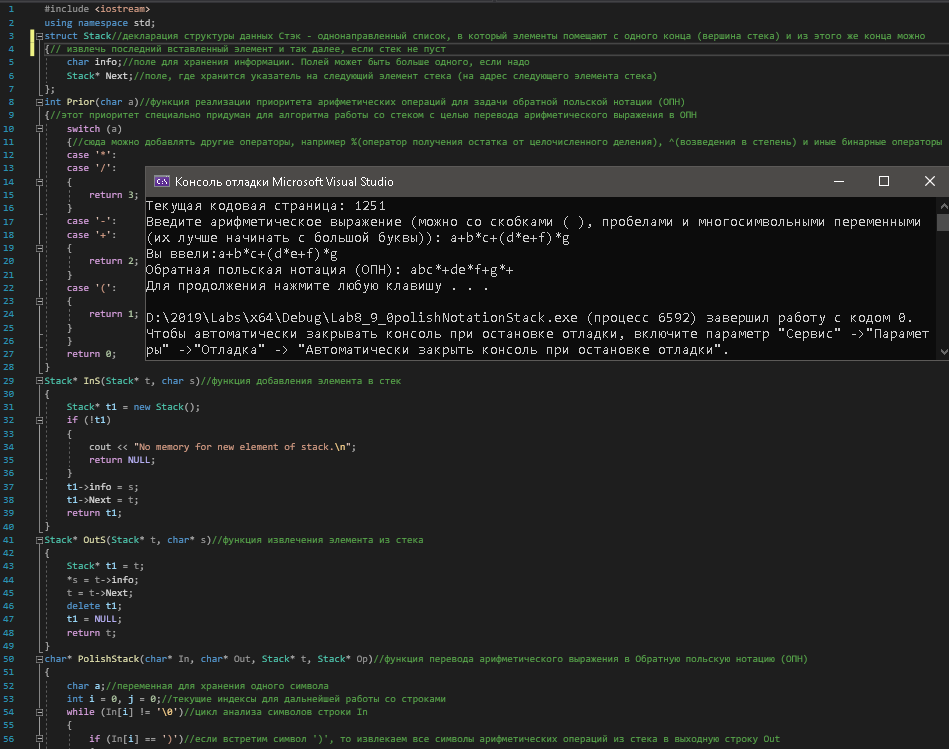
|  |  |
| --- | --- |
|  | В = ”abс\*+de\*f+g” |
| \* |  |
| + |  |

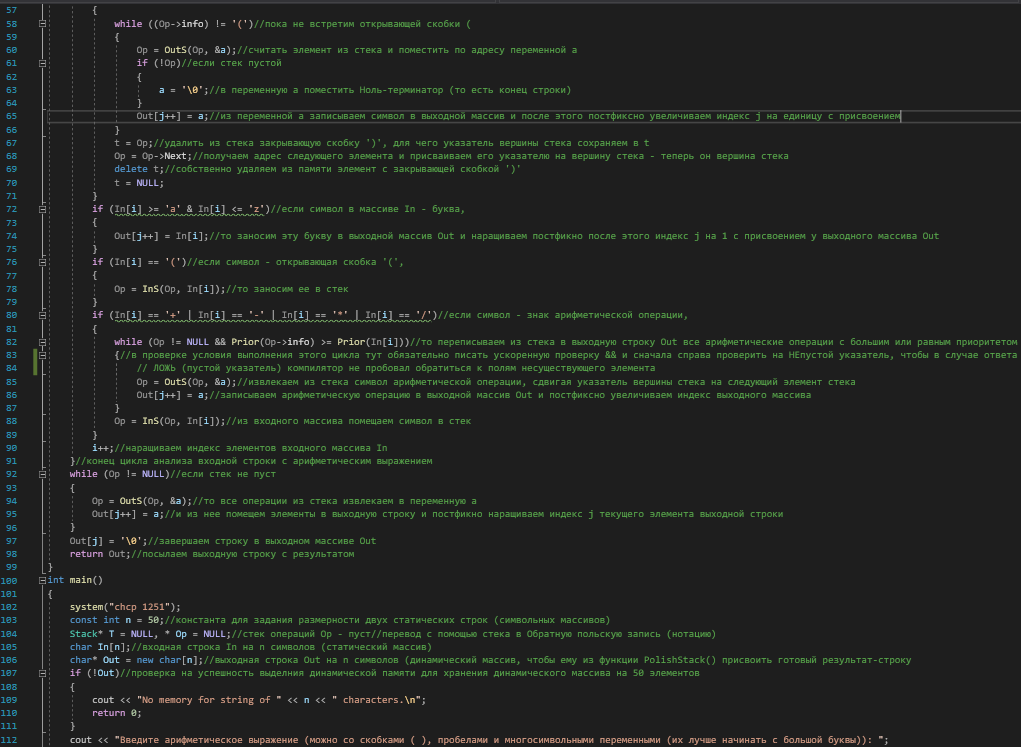
Все символы строки S рассмотрены, следовательно, анализируем состояние стека, если он не пуст, то переписываем все его элементы в строку В:

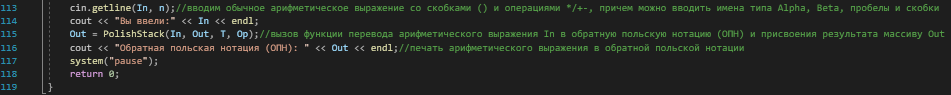
|  |  |
| --- | --- |
|  | В = ”abс\*+de\*f+g\*+” |

Таким образом, просмотрев исходную информацию только один раз, мы решили поставленную задачу.

1. **Пример выполнения программы**







1. **Задания по вариантам**

**Задание 1**

Написать программу по созданию, просмотру, добавлению и решению поставленной задачи для однонаправленного линейного списка (стек и/или очередь):

1. Создать список из случайных целых чисел, лежащих в диапазоне от –50 до +50 и преобразовать его в два списка. Первый должен содержать только положительные числа, а второй – только отрицательные. Порядок следования чисел должен быть сохранен.

2. Создать список из случайных целых чисел и удалить из него записи с четными числами.

3. Создать список из случайных положительных и отрицательных целых чисел (от –10 до 10) и удалить из него отрицательные элементы.

4. Создать список из случайных целых чисел и поменять местами крайние элементы.

5. Создать список из случайных целых чисел и удалить элементы, заканчивающиеся на цифру 5.

6. Создать список из случайных целых чисел и поменять местами элементы, содержащие максимальное и минимальное значения.

7. Создать список из случайных целых чисел. Перенести в другой список все элементы, находящиеся между вершиной и элементом с максимальным значением.

8. Создать список из случайных целых чисел. Перенести в другой список все элементы, находящиеся между вершиной и элементом с минимальным значением.

9. Создать список из случайных чисел, определить количество элементов, находящихся между минимальным и максимальным элементами, и удалить их.

10. Создать список из случайных чисел и определить количество элементов, имеющих значения, меньше среднего значения от всех элементов, и удалить эти элементы.

11. Создать список из случайных чисел, вычислить среднее арифметическое и заменить им первый элемент.

12. Создать список из случайных целых чисел, разделить его на два: в первый поместить все четные, а во второй – нечетные числа.

13. Создать список из случайных целых чисел в диапазоне от 1 до 10, определить наиболее часто встречающееся число и удалить его.

14. Создать список из случайных целых чисел и удалить из него каждый второй элемент.

15. Создать список из случайных целых чисел и удалить из него каждый нечетный элемент.

**Задание 2**

Написать программу по созданию, просмотру, добавлению и решению поставленной задачи для двунаправленного линейного списка.

1. Создать список из случайных целых чисел. Найти минимальный элемент и сделать его первым.

2. Создать два списка из случайных целых чисел. В первом найти максимальный элемент и за ним вставить элементы второго.

3. Создать список из случайных целых чисел. Удалить из списка все элементы, находящиеся между максимальным и минимальным элементами.

4. Упорядочить элементы списка случайных целых чисел в порядке возрастания.

5. Создать список из случайных целых чисел. Удалить из списка все элементы, находящиеся до максимального элемента.

6. Создать список из случайных целых чисел. Удалить из списка все элементы, находящиеся после минимального элемента.

7. Создать список из случайных целых чисел. Из элементов, расположенных между максимальным и минимальным элементами, создать второй список, а из остальных – третий.

8. Создать список из случайных положительных и отрицательных целых чисел. Образовать из него два списка, первый должен содержать отрицательные числа, а второй – положительные.

9. Создать список из случайных целых чисел. Удалить из списка все элементы, находящиеся после максимального элемента.

10. Создать два списка из случайных целых чисел. Вместо элементов первого списка, заключенных между максимальным и минимальным элементами, вставить второй список.

11. Создать список из случайных целых чисел. Удалить из списка элементы с повторяющимися более одного раза значениями.

12. Создать список из случайных целых чисел и удалить все элементы, кратные 5.

13. Создать список из случайных целых чисел. Удалить из списка все элементы, большие среднего арифметического.

14. Создать список из случайных чисел. Преобразовать его в кольцо. Предусмотреть возможность движения по кольцу в обе стороны с отображением места положения текущего элемента.

15. Создать список из случайных целых чисел. Удалить из списка все элементы, находящиеся между максимальным и минимальным элементами.

**Задание 3**

Написать программу формирования обратной польской записи и расчета полученного выражения. Предусмотреть возможности того, что идентификаторы могут состоять более чем из одного символа и могут быть использованы операции % и возведение в степень**^**. Можно использовать стек или деревья. Результат работы программы проверить на конкретном примере по своему варианту.

Например, если ввести выражение (a + b)\*(c – d)/e и значения переменных а = 3, b = 5, c = 6, d = 9, е = 7, то должны получиться следующие результаты:

Постфиксная форма: ab+cd– \*e/

Результат расчета: – 3.42857

Таблица с вариантами выражений и проверочных значений данных:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Выражение** | **a** | **b** | **c** | **d** | **e** | **Результат** |
| 1 | a/(b– c)\*(d+e) | 8.6 | 2.4 | 5.1 | 0.3 | 7.9 | – 26.12 |
| 2 | (a+b)\*(c– d)/e | 7.4 | 3.6 | 2.8 | 9.5 | 0.9 | – 81.89 |
| 3 | a– (b+c\*d)/e | 3.1 | 5.4 | 0.2 | 9.6 | 7.8 | 2.16 |
| 4 | a/b– ((c+d)\*e) | 1.2 | 0.7 | 9.3 | 6.5 | 8.4 | – 131.006 |
| 5 | a\*(b– c+d)/e | 9.7 | 8.2 | 3.6 | 4.1 | 0.5 | 168.78 |
| 6 | (a+b)\*(c– d)/e | 0.8 | 4.1 | 7.9 | 6.2 | 3.5 | 2.38 |
| 7 | a\*(b– c)/(d+e) | 1.6 | 4.9 | 5.7 | 0.8 | 2.3 | – 0.413 |
| 8 | a/(b\*(c+d))– e | 8.5 | 0.3 | 2.4 | 7.9 | 1.6 | 1.151 |
| 9 | (a+(b/c– d))\*e | 5.6 | 7.4 | 8.9 | 3.1 | 0.2 | 0.666 |
| 10 | a\*(b+c)/(d– e) | 0.4 | 2.3 | 6.7 | 5.8 | 9.1 | – 1.091 |
| 11 | a– (b/c\*(d+e)) | 5.6 | 3.2 | 0.9 | 1.7 | 4.8 | – 17.51 |
| 12 | (a– b)/(c+d)\*e | 0.3 | 6.7 | 8.4 | 9.5 | 1.2 | – 0.429 |
| 13 | a/(b+c– d\*e) | 7.6 | 4.8 | 3.5 | 9.1 | 0.2 | 1.173 |
| 14 | a\*(b– c)/(d+e) | 0.5 | 6.1 | 8.9 | 2.4 | 7.3 | – 0.144 |
| 15 | (a+b\*c)/(d– e) | 9.1 | 0.6 | 2.4 | 3.7 | 8.5 | – 2.196 |

1. **Контрольные вопросы**
2. Что такое «обратная польская запись» и зачем она нужна?
3. С помощью каких динамических структур данных можно построить ОПН?
4. В чем суть алгоритма Дейкстры для построения ОПН?
5. Что такое «стековый приоритет операций»?

**Литература**

**Дейтел,** Х.М. Как программировать на С++ / Х.М. Дейтел, П.Дж. Дейтел . – М. : Бином-Пресс , 2018 . – 1456 с.

**Павловская**, Т.А. С++. Объектно-ориентированное программирование : практикум / Т.А. Павловская, Ю.А. Щупак . – СПб. : Питер , 2019 . – 265 с.

**Страуструп**, Б. Язык программирования С++ / Б. Страуструп . – СПб. : Бином-Пресс , 2019 . – 1054 с.

Преподаватель Шаляпин Ю.В.

|  |
| --- |
| Рассмотрено на заседании цикловой  комиссии ПОИТ № 10  Протокол №\_\_\_\_от «\_\_\_»\_\_\_\_\_\_\_\_\_\_2021 г.  Председатель ЦК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |