МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Отчет по выполнению практического задания №8
Тема: Нотации выражений
Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Антонов А.Д.

Группа ИКБО-01-20

Москва 2021

Оглавление

1. Задание 1 (Вариант №1)

1.1. Процесс выполнения упражнения 1

Условие задания:

Провести преобразование инфиксной записи выражения в префиксную нотацию, расписывая процесс по шагам для выражения: *S=a+(b-c*k)/(d*e-f)*

При переводе в префиксную запись выражение читается справа налево, а формируется в обратном порядке, при этом каждый новый символ добавляется в начало.

Таблица 1. Алгоритм преобразования выражения S

Символ	Стек	Вывод	Комментарий
)	)		Оператор добавляется в стек
f	)	f	Операнд добавляется в начало
-	-)	f	Оператор с наивысшим приоритетом добавляется в стек
e	-)	ef	Операнд добавляется в начало
*	-*)	ef	Оператор с наивысшим приоритетом добавляется в стек
d	-*)	def	Операнд добавляется в начало
(		-*def	Операторы выталкиваются до открывающей скобки
/	/	-*def	Оператор добавляется в стек
)	)/	-*def	Оператор добавляется в стек
k	)/	k-*def	Операнд добавляется в начало
*	*)/	k-*def	Оператор с наивысшим приоритетом добавляется в стек
c	*)/	ck-*def	Операнд добавляется в начало
-	-)/	*ck-*def	- имеет меньший приоритет, поэтому выталкивается *, после добавляется -
b	-)/	b*ck-*def	Операнд добавляется в начало
(	/	-b*ck-*def	Операторы выталкиваются внутри скобок
+	+	/-b*ck-*def	+ имеет меньший приоритет, поэтому выталкивается /, после добавляется -

a	+	a/-b*ck-*def	Операнд добавляется в начало

В конце добавляем из стека оставшиеся операторы:

$S = + a / - b * c k - * d e f$

1.2. Процесс выполнения упражнения 2

Условие задания:

Представить инфиксную нотацию выражений (идентификаторы односимвольные)

- x y z * a b d / - + c - +
- x y z * d / + a b * c o k - / - -

Таблица 2. Алгоритм преобразования выражения 1

Выражение	Комментарий
x y z * a b d / - + c - +	Идём по выражению слева направо до первого оператора — *, далее применяем его к двум операндам, стоящим справа от него
x y*z a b d / - + c - +	Аналогично первому шагу находим операнд / применяем его аналогично. Действуем так, пока все операторы не встанут на свои места
x y*z a b/d - + c - +	Прим. *операторы и операнды,которые поставлены на свои места записываются без пробела*
x y*z a-b/d + c - +	
x y*z+a-b/d c - +	
x y*z+a-b/d-c +	
x+y*z+a-b/d-c	

Инфиксная запись выражения 1: $S_1 = x + y*z + a - b/d - c$

Аналогично первому выражению преобразуем второе

Таблица 2. Алгоритм преобразования выражения 2

Выражение	Комментарий
x y z * d / + a b * c o k - / - -	
x y*z d / + a b * c o k - / - -	
x y*z/d + a b * c o k - / - -	
x+y*z/d a b * c o k - / - -	
x+y*z/d a*b c o k - / - -	
x+y*z/d a*b c o-k / - -	
x+y*z/d a*b c o-k / - -	
x+y*z/d a*b c/(o-k) - -	
x+y*z/d a*b-c/(o-k) -	
x+y*z/d-a*b-c/(o-k)	

Инфиксная запись выражения 2: $S_2 = x + y * z / d – a * b – c / (o – k)$

1.3. Процесс выполнения упражнения 3

Условие задания:

Представить префиксную нотацию выражений, указанных в пункте 1.2.

Таблица 4. Алгоритм преобразования выражения 1

Символ	Стек	Вывод
c		c
-	-	
d	-	dc
/	/-	dc
b	/-	bdc
-	-	-/bdc
a	-	a-/bdc
+	+	-a-/bdc
z	+	z-a-/bdc
*	*+	z-a-/bdc
y	*+	yz-a-/bdc
+	+	*+ yz-a-/bdc
x	+	x*+ yz-a-/bdc
		+ x*+ yz-a-/bdc

Таблица 4. Алгоритм преобразования выражения 2

Символ	Стек	Вывод
)	)	
k	)	k
-	-)	k
o	-)	ok
(		-ok
/	/	-ok
c	/	c-ok
-	-	/c-ok
b	-	b/c-ok
*	*-	b/c-ok
a	*-	ab/c-ok
-	-	*- ab/c-ok
d	-	d*- ab/c-ok
/	/-	d*- ab/c-ok
z	/-	zd*- ab/c-ok
*	*	/-zd*- ab/c-ok
y	*	y/- zd*- ab/c-ok
+	+	*y/-zd*-ab/c-ok
x	+	x*y/-zd*-ab/c-ok
		+ x*y/-zd*-ab/c-ok

1.4. Процесс выполнения упражнения 4

Условие задания:

Провести вычисление значения выражения, представленного в постфиксной форме, расписывая процесс по шагам: 7 2 3*5 8 2/-+1-+

Таблица 6. Процесс вычислений

Выражение	Вычисление
7 2 3*5 8 2 / - + 1 - +	
7 2*3 5 8 2 / - + 1 - +	2*3=6
7 6 5 8/2 - + 1 - +	8/2=4
7 6 5-4 + 1 - +	5-4=1
7 6+1 1 - +	6+1=7
7 7-1 +	7-1=6
7+6	7+6=13
13	

Значение выражения = 13.

2. Программная реализация задания 2

2.1. Разработать функцию (функции) преобразования инфиксной формы выражения в постфиксную форму.

2.1.1. Постановка задачи:

Выбрать структуру данных для реализации стека и описать функции работы со стеком, реализовать функцию, преобразования инфиксной формы выражения в постфиксную форму.

2.1.2. Код программы:

```
#include <iostream>
#include <string>

using namespace std;

class Node {
public:
    Node *pNext;
    char data;
    int pr;

    Node (char data, Node *pNext = nullptr);
    void setPr (char i);
};
```

```
class LinkedList {
private:
    Node *head;
    int size;

public:
    LinkedList();
    void push_front(char data);
    ~LinkedList() = default;
    char pop_front();
    int getSize() const { return this->size; }
    int getPr();
};
```

```
16  Node::Node(char data, Node *pNext) {
17      this->data = data;
18      this->setPr(data);
19      this->pNext = pNext;
20  }
21
22  void Node::setPr(char i) {
23      switch (i) {
24          case '(': case ')': { pr = 1; break; }
25          case '-': case '+': { pr = 2; break; }
26          case '*': case '/': { pr = 3; break; }
27          default: { pr = 0; break; }
28      }
29  }
```

```
45  LinkedList::LinkedList() {
46      size = 0;
47      head = nullptr;
48  }
49
50  void LinkedList::push_front(char data) {
51      head = new Node(data, head);
52      size++;
53  }
54
55  char LinkedList::pop_front() {
56      Node *temp = head;
57      head = head->pNext;
58      size--;
59      return (temp->data);
60  }
61
62  int LinkedList::getPr() {
63      return (head->pr);
64  }
```

```
66  void infToPref (const string& s, LinkedList &L) {
67      string result;
68      int p;
69      for (char i : s) {
70          if (isalpha(i)) { result += i; continue; }
71          switch (i) {
72              case '(': case ')': { p = 1; break; }
73              case '-': case '+': { p = 2; break; }
74              case '*': case '/': { p = 3; break; }
75              default: { p = 0; break; }
76          }
77          if ((i != ' ') && (i != '(')) {
78              if (i == ')')
79                  while (L.getSize() != 0)
80                      result += L.pop_front();
81              else
82                  if (L.getSize() == 0 || p > L.getPr())
83                      L.push_front(i);
84                  else {
85                      while (L.getSize() != 0)
86                          result += L.pop_front();
87                      L.push_front(i);
88                  }
89          }
90      }
91      while (L.getSize() != 0)
92          result += L.pop_front();
93      cout << "The result: " << result;
94  }
```

```
100 int main() {
101     LinkedList L;
102     string s;
103     cout << "Input string: ";
104     getline(cin, s);
105     infToPref(s, L);
106 }
```

2.1.3. Тестирование:

Номер теста	Входные данные	Ожидаемый результат	Результат программы
1	x+y*z+a-b/d-c	xyz*+a+bd/-c-	Input string: x+y*z+a-b/d-c The result: xyz*+a+bd/-c-
2	x+y*z/d-a*b-c/o	xyz*+d/ab*-co/-	Input string: x+y*z/d-a*b-c/o The result: xyz*+d/ab*-co/-
3	a+b/c+d+e-f/m*k	abc/+d+e+fm/-k*	Input string: a+b/c+d+e-f/m*k The result: abc/+d+e+fm/-k*

2.2. Реализовать операции стек: втолкнуть элемент в стек, вытолкнуть элемент из стека, вернуть значение элемента в вершине стека, сделать стек пустым, определить пуст ли стек.

2.2.1. Постановка задачи:

- Создать класс или просто заголовочный файл с функциями.
- Применить операции для вычисления значения выражения п.4 данного варианта.

Так как стек реализован с помощью односвязного списка, код для него можно взять из предыдущих практик. Единственный новый метод – определение пуст ли стек.

2.2.2. Код программы

Код для нового метода:

```
98  bool LinkedList::isEmpty() const
99  {
100     if (this->getSize() == 0)
101         return false;
102     else
103         return true;
104 }
```

Алгоритм main() для проверки работы функции isEmpty:

```
117 ▶ int main() {
118       LinkedList L;
119       for(int i = 0; i < 2; i++) {
120           if (L.isEmpty())
121               cout << i + 1 << ": True" << endl;
122           else cout << i + 1 << ": False" << endl;
123           L.push_back( i: 'a');
124       }
125   }
```

```
1: False
2: True
```

Рис.1 Результат работы программы

Из рис. 1 мы видим, что функция isEmpty работает корректно.

Для реализации вычисления значения выражения п.4 данного варианта напишем функцию, которая будет вычислять значение выражения в постфиксной записи:

```
void postEqual (string s) {
    int op1 = 0, op2 = 0;
    while ((s.size() != 2) && (s.size() != 1)) {
        for (int i = 0; i < s.length(); i++)
            if (isdigit(s[i])) {
                if (i == 0) { op2 = s[i] - '0'; }
                else { op1 = op2; op2 = s[i] - '0'; }
            }
            else {
                switch (s[i]) {
                    case '+': { op2 = op1 + op2; break; }
                    case '-': { op2 = op1 - op2; break; }
                    case '*': { op2 = op1 * op2; break; }
                    case '/': { op2 = op1 / op2; break; }
                    default: break;
                }
                switch (s[i]) {
                    case '+': case '-': case '*': case '/': {
                            s.insert(i + 1, to_string(op2));
                            s.erase(i - 4, 5);
                            i = -1;
                            cout << s << endl;
                            break;
                    }
                    default: break;
                }
            }
        cout << "The result: " << s << endl;
    }
}
```

Алгоритм функции main():

```
int main()
{
    string s;
    cout << "Input string: ";
    getline(cin, s);
    postEqual(s);
}
```

```
Input string: 7 2 3 * 5 8 2 / - + 1 - +
7 6 5 8 2 / - + 1 - +
7 6 5 4 - + 1 - +
7 6 1 + 1 - +
7 7 1 - +
7 6 +
13
The result: 13
```

Рис.2 Результат работы программы

Результат программы совпадает с вычислениями, сделанными нами в пункте 1.4, следовательно, программа работает корректно.

2.3. Реализация структуры из двух стеков внутри одного массива

2.3.1. Постановка задачи:

Реализовать стек на следующей структуре: хранить два стека в одном массиве, когда один располагается в начале массива и растет к концу массива, а второй располагается в конце и растет к началу. Реализуйте операцию Push(x,S) –втолкнуть элемент x в стек S, где S один или другой стек.

Так как стек реализован с помощью односвязного списка, код для него можно взять из предыдущих практик. Процесс добавления и вывода всех элементов реализуем через меню.

2.3.2. Код функции и программы:

```
140  void push(char x, LinkedList &L, int &a) {
141      switch (a) {
142          case 1: { L.push_back(x); break; }
143          case 2: { L.push_front(x); break; }
144      }
145  }
```

```cpp
int main() {
    LinkedList L1,L2;
    int a; char x;
    vector<char> v;
    cout << "Menu:" << endl
         << "Press 1 to add element at stack 1" << endl
         << "Press 2 to add element at stack 2" << endl
         << "Press 3 to show 2 stacks and array" << endl
         << "Press 0 to exit" << endl;
    while (true) {
        cout << "Enter number: "; cin >> a;
        switch (a) {
            case 0:
                { cout << "Closing program..."; return 0; }
            case 1: {
                cout << "Input x: "; cin >> x;
                push(x, L1,a); break;
            }
            case 2: {
                cout << "Input x: "; cin >> x;
                push(x, L2,a); break;
            }
            case 3: {
                cout << "Stack 1: "; L1.print_front(L1);
                cout << "Stack 2: "; L2.print_back(L2);
                while (L1.getSize()!= 0)
                    v.push_back(L1.pop_front());
                while (L2.getSize() != 0)
                    v.push_back(L2.pop_front());
                cout << "All array: ";
                for (int i = 0; i < v.size(); i++)
                    cout << v[i] << " ";
                cout << endl; break;
            }
            default: break;
        }
    }
}
```

Рис.3 Результат работы программы

Из рис.3 мы видим, что программа работает корректно.

Выводы

В ходе данной практической работы была разработана ещё одна динамическая структура данных – стек. За основу была взята другая динамическая структура данных, рассмотренная в одной из прошлых практических работ – односвязный список. Были разработаны методы для работы со стеком: метод для вставки нового элемента в стек, метод для извлечения элемента из стека, метод для проверки стека на пустоту, метод (деструктор) для очисти стека и освобождения выделенной памяти. Были изучены постфиксная и префиксная нотации, а также вычислены значения выражений, записанных в виде этих нотаций. Была разработана функция для сложения больших чисел, выходящих за рамки стандартных форматов. Все разработанный функции и методы прошли проверку на корректность результата.

Список информационных источников

1. Лекционный материал по структурам и алгоритмам обработки данных Алпатова А.Н.
2. Math.semestr – Обратная польская запись - https://www.wikidocs.ru/preview/59279/11 (дата обращения: 29.05.2021)
3. YouTube – Реализация односвязного списка C++ на канале #SimpleCode (Уроки №133-135) - https://www.youtube.com/watch?v=SajrPhE6FoQ (дата обращения: 25.04.2021).