Лабораторная работа №16

Динамические структуры данных

Оглавление

Цель

Научиться работать с динамическими структурами данных на примере стека и очереди, реализованных с помощью односвязного списка.

Инструкция:

Рассмотрим два методических примера.

Пример 1. Записать в стек последовательно числа от 0 до 9. Извлечь из стека и вывести на экран.

main.cpp | stack.cpp | stack.h

```
#include <iostream>
#include "stack.h"
using namespace std;



int main()
{
    int i;
    Node* Top=NULL;      //вершина стека
    int Count=0;         //количество элементов в стеке

//добавление элементов в стек

    for(int i=0;i<10;i++)
        {
            Push(&Top,Count,i);
            cout<<"| "<<Count <<" | "<<i<<" |"<<endl;
        }

        cout<<"---------------"<<endl;

//добавление элементов в стек

    while(Count>0)
        {
            Pop(&Top,Count,i);
            cout<<"| "<<Count <<" | "<<i<<" |"<<endl;
        }

    return 0;
}

```

main.cpp | stack.cpp | stack.h

```cpp
#include "stack.h"

//функция добавления элемента в стек
bool Push(Node **Top,int& Count,int data)
{
    if(*Top==NULL)
        {
            *Top=new Node;
            (*Top)->prev=NULL;
        }
    else
        {
            Node *temp;
            temp=new Node;
            temp->prev=*Top;
            *Top=temp;
        }

    Count++;
    (*Top)->data=data;

return true;
}
//функция извлечения элемента из стека
bool Pop(Node **Top,int& Count,int& data)
{
    if(*Top==NULL) return false;

         Node* temp=(*Top)->prev;
         data=(*Top)->data;
         delete *Top;
         *Top=temp;
         Count--;

        return true;
}

```

main.cpp | stack.cpp | stack.h

```
1  #ifndef STACK_H_INCLUDED
2  #define STACK_H_INCLUDED
3
4  //описание узла стека
5  struct Node{
6      int data;
7      Node* prev;
8  };
9
10 extern Node* Top;    //вершина стека
11 extern int Count;    //количество элементов в стеке
12
13 bool Push(Node **Top,int& Count,int data);
14 bool Pop(Node **Top,int& Count,int& data);
15 #endif // STACK_H_INCLUDED
16
```

Пример 2. Записать в очередь последовательно числа от 0 до 9. Извлечь из очереди и вывести на экран.

main.cpp | qu.cpp | qu.h

```cpp
#include <iostream>
#include "qu.h"
using namespace std;

int main()
{
    int i;
    Node* First=NULL;    // первый элемент очереди
    int Count=0;         //количество элеменов в очереди

//добавление элементов в очередь
    for(i=1;i<=10;i++)
        {
            Push(&First,Count,i);
            cout<<"| "<<Count <<" | "<<i<<" |"<<endl;
        }

        cout<<"--------------"<<endl;
//извлечение элементов из очереди
    while(Count>0)
        {
            Pop(&First,Count,i);
            cout<<"| "<<Count <<" | "<<i<<" |"<<endl;
        }
 return 0;
}
```

main.cpp | qu.cpp | qu.h

```cpp
#ifndef QU_H_INCLUDED
#define QU_H_INCLUDED

struct Node{
    int data;
    Node* next;
};

extern Node* First;
extern int Count;

bool Push(Node **First,int& Count,int data);
bool Pop(Node **First,int& Count,int& data);

#endif // QU_H_INCLUDED

```

main.cpp | qu.cpp | qu.h

```cpp

#include "qu.h"
//функция добавления элемента в очередь
bool Push(Node **First,int& Count,int data)
{
    if(*First==NULL)
        {
            *First=new Node;
            (*First)->next=NULL;
            (*First)->data=data;
            Count=1;
        }
    else
        {
            Node *temp;
            temp=*First;
            while(temp->next!=NULL)
                {
                   temp=temp->next;
                }
            temp->next=new Node;
            temp->next->data=data;
            temp->next->next=NULL;
           Count++;
        }
return true;
}
//функция извлечения элемента из очереди
bool Pop(Node **First,int& Count,int& data)
{
    if(*First==NULL) return false;

         Node* temp=(*First)->next;
         data=(*First)->data;
         delete *First;
         *First=temp;
         Count--;

        return true;
}
```

Задания для самостоятельного выполнения:

Наберите код программы из примера 1 и примера 2. Проанализируйте результат.

Домашнее задание:

БЛОК А

Задание на отметку «удовлетворительно»

С экрана считываются числа до тех пор, пока не будет введен ноль. Четные числа поместите в стек, а нечетные – в очередь. Выведите содержимое стека и очереди.

БЛОК В

Задания на отметку «хорошо»

1. Используя стек, напечатать символы некоторой величины строкового типа в обратном порядке.

   С использованием стека задача решается следующим образом. Каждый символ заданной величины, начиная с первого, размещаем в стеке. В результате на вершине стека окажется последний символ, "под" ним — предпоследний и т.д. Извлекая символы из стека, выводим их на экран в порядке, обратном исходному.

2. Написать программу, которая определяет, является ли введенная скобочная структура правильной. Примеры правильных скобочных выражений: (), (())(), ()(), ((())), неправильных —)(, ())((), (,)))), (((().

   Можно рассуждать так. Рассматриваем последовательно каждый символ заданной величины слева направо. Если очередной символ — левая скобка, то размещаем ее в стеке, если правая — то извлекаем элемент из стека (это обязательно должна быть левая скобка). После рассмотрения всей строки, если все правильно, стек должен оказаться пустым.

3. Первая строка входного файла содержит единственное натуральное число n – количество запросов.

   Далее записаны n чисел от -1 до 1000000, разделенных пробелами и/или переводами строк – запросы. Если запрос равен -1, следует снять элемент со стека (если он не пуст). Иначе следует положить этот элемент в стек

   Формат выходных данных: в файл для каждого запроса -1 выведите убираемый элемент, либо -1, если стек уже пуст.

   Пример. Вход:
   7
   2 3 -1 9 -1 -1 – 1
   Выход:
   3 9 2 -1

4. Рассмотрим запись арифметических выражений, в которых сначала следуют два операнда арифметической операции, а затем знак операции: 3 4 + (эквивалентно 3+4), 2 3 4 5 6 * + - / (эквивалентно 2 / (3 - (4 + (5 * 6)))).Эта нотация записи выражений называется обратной

польской нотацией. Напишите программу - калькулятор арифметических выражений записанных в обратной польской нотации.

Каждое следующее число помещается в стек. Если встречается знак операции, то два числа из стека, для них вычисляется значение соответствующей бинарной арифметической операции, и результат помещается в стек.

5. Написать программу, которая определяет, является ли введенная скобочная структура правильной. Примеры правильных скобочных выражений: <>, <<>><>, <><>, <<<>>>, неправильных — ><, <>><<>, <, >>>>, <<<>>.

Можно рассуждать так. Рассматриваем последовательно каждый символ заданной величины слева направо. Если очередной символ — левая скобка, то размещаем ее в стеке, если правая — то извлекаем элемент из стека (это обязательно должна быть левая скобка). После рассмотрения всей строки, если все правильно, стек должен оказаться пустым.

6. Написать программу, которая определяет, является ли введенная скобочная структура правильной. Примеры правильных скобочных выражений: [], [[]][], [][], [[[]]], неправильных —][, []][[], [,]]]], [[[].

Можно рассуждать так. Рассматриваем последовательно каждый символ заданной величины слева направо. Если очередной символ — левая скобка, то размещаем ее в стеке, если правая — то извлекаем элемент из стека (это обязательно должна быть левая скобка). После рассмотрения всей строки, если все правильно, стек должен оказаться пустым.

7. Написать программу, которая определяет, является ли введенная скобочная структура правильной. Примеры правильных скобочных выражений: {}, {{}}{}, {}{}, {{{}}}, неправильных — }{, {}}{{}, {, }}}}, {{{}.

Можно рассуждать так. Рассматриваем последовательно каждый символ заданной величины слева направо. Если очередной символ — левая скобка, то размещаем ее в стеке, если правая — то извлекаем элемент из стека (это обязательно должна быть левая скобка). После рассмотрения всей строки, если все правильно, стек должен оказаться пустым.

8. Описать структурный тип автомобиль (номер, марка, цвет). Заполнить авотмобилями «стоянку» по принципу стека. Чтобы машина могла выехать со стоянки, должны выехать все стоящие между ней и выходом машины (вверх по стеку), затем все «отъехавшие» машины возвращаются обратно.

9. Используя стек, из некоторого целого числа получить новое число, которое получается из исходного перестановкой цифр в обратном порядке.

С использованием стека задача решается следующим образом. Каждую цифру исходного числа, начиная с первого, размещаем в стеке. В результате на вершине стека окажется последняя цифра, "под" ней — предпоследняя и т.д. Извлекая символы из стека, выводим их на экран в порядке, обратном исходному.

10. Используя стек, из некоторого вещественного числа получить новое число, которое получается из исходного перестановкой цифр в обратном порядке. В результате должно получиться число вещественного типа.

С использованием стека задача решается следующим образом. Каждую цифру исходного числа, начиная с первого, размещаем в стеке. В результате на вершине стека окажется последняя цифра, "под" ней — предпоследняя и т.д. Извлекая символы из стека, выводим их на экран в порядке, обратном исходному.

11. Дана величина *a* строкового типа из четного количества символов. Получить и напечатать величину *b*, состоящую из символов первой половины величины *a*, записанных в обратном порядке, после которых идут символы второй половины величины *a*, также записанные в обратном порядке. Например, при *a* = "привет" *b* должно быть равно "ипртев".

12. Дана величина *a* строкового типа из четного количества символов. Получить и напечатать величину *b*, состоящую из символов первой половины величины *a*, записанных в обратном порядке, после которых идут символы второй половины величины *a*, записанные в прямом порядке. Например, при *a* = "привет" *b* должно быть равно "ипрвет".

13. С экрана вводится строка. Используя стек, получить новую строку из исходной, убирая все одинаковые рядом стоящие символы.
Пример: вводится «aabbcdaaaa», в результате должно получиться: «abcda».

14. Дан набор из 10 чисел. Создать две очереди: первая должна содержать числа из исходного набора с нечетными номерами (1, 3, ..., 9), а вторая — с четными (2, 4, ..., 10); порядок чисел в каждой очереди должен совпадать с порядком чисел в исходном наборе. Вывести указатели на начало и конец каждой из полученных очередей.

15. Дан набор из 10 чисел. Создать две очереди: первая должна содержать положительные числа из исходного набора, а вторая — отрицательные; порядок чисел в каждой очереди должен совпадать с порядком чисел в исходном наборе. Вывести указатели на начало и конец каждой из полученных очередей.

16. Дана последовательность ненулевых целых чисел. Признаком конца последовательности является число 0. Найдите среди них первый наибольший отрицательный элемент. Если такого элемента нет, то выведите сообщение об этом. Для решения используйте очередь.

17. Дана последовательность ненулевых целых чисел. Признаком конца последовательности является число 0. Найдите среди них первый наибольший четный элемент. Если такого элемента нет, то выведите сообщение об этом. Для решения используйте очередь.

18. За один просмотр файла действительных чисел напечатать элементы файла в следующем порядке: сначала – все числа, меньшие a, затем – все числа из отрезка [a, b], и наконец – все остальные числа, сохраняя исходный порядок в каждой из этих трех групп чисел. Числа a и b задает пользователь. Использовать очереди.

19. За один просмотр файла действительных чисел напечатать элементы файла в следующем порядке: сначала – все отрицательные числа, а затем –все положительные, сохраняя исходный порядок в каждой из групп чисел. Использовать очереди.

20. Написать программу, которая циклически выводит через пробел ваше имя со смещением на один символ (бегущая строка), до тех пор, пока следующим не должно будет выводится исходное слово еще раз. Использовать очередь.
Пример: введено - IVAN результат – IVANVANIANIVNIVA

БЛОК С

Задания на отметку «отлично»

1. Рассмотрим запись арифметических выражений, в которых сначала следуют два операнда арифметической операции, а затем знак операции: 3 4 + (эквивалентно 3+4), 2 3 4 5 6 * + - / (эквивалентно 2 / (3 - (4 + (5 * 6)))).Эта нотация записи выражений называется обратной польской нотацией. Напишите программу - калькулятор арифметических выражений записанных в обратной польской нотации. Предусматривать «защиту от дурака» (например, ситуацию, когда введено выражение с избыточным числом операций, что приводит к преждевременному опустошению стека, или избыточное число операндов в выражении (в конце в стеке более 1 числа))

   Каждое следующее число помещается в стэк. Если встречается знак операции, то два числа из стека, для них вычисляется значение соответствующей бинарной арифметической операции, и результат помещается в стек.

2. Реализовать стек, моделирующий поступление товаров на склад и их продажу. Для этого описать структурный тип «партия товара» (количество единиц товара, цена единицы товара). Описать функцию добавления новой партии, функцию отгрузки (продажи) партии (при этом отгружается не обязательно столько единиц товара, сколько хранится в крайней партии), функцию вывода текущего состояния склада.

3. Составить программу, отыскивающую проход по лабиринту. Лабиринт представляется в виде матрицы, состоящей из квадратов. Каждый квадрат либо открыт, либо закрыт. Вход в закрытый квадрат запрещен. Если квадрат открыт, то вход в него возможен со стороны, но не с угла. Каждый квадрат определяется его координатами в матрице. Программа находит проход через лабиринт, двигаясь от заданного входа. После отыскания прохода программа выводит найденный путь в виде координат квадратов. Для хранения пути использовать стек.

4. Написать программу вычисления некоторого арифметического выражения, содержащего круглые скобки и знаки сложения, используя стек. Выражения вводится с экрана как строка. Выполнять проверку, а правильно ли записано выражение.

5. С экрана вводится строка, представляющая собой скобочное выражение. Необходимо проверить является ли это скобочное выражение верным, т.е. в правильном ли порядке закрываются все открывающиеся скобки. Типы открывающихся и закрывающих их скобок вводятся с экрана. Если выражение не верное - выводите порядковый номер первого символа (скобки), нарушающего правильность расстановки скобок.

   Пример выполнения программы:

   Введите количество типов скобок<< ***2***
   Введите первый тип открывающейся скобки<< ***(***
   Введите первый тип закрывающейся скобки<< ***)***
   Введите второй тип открывающейся скобки<< ***{***
   Введите первый тип закрывающейся скобки<< ***}***
   Введите выражение: ***(){}){***
   Выражение ***не верно****, ошибка в* ***7*** *скобке*

6. Из заданного текста перенести все цифры в конец каждой строки, сохранив их порядок. Использовать очереди.
7. Из заданного текста перенести все цифры в конец каждой строки, в обратном порядке. Использовать стек.
8. Из заданного текста перенести все буквы в конец каждой строки, сохранив их порядок. Использовать очереди.

9. Из заданного текста перенести все буквы в конец каждой строки, в обратном порядке. Использовать стек.
10. Из заданного текста перенести все знаки препинания в конец каждой строки, сохранив их порядок. Использовать очереди.
11. Из заданного текста перенести все знаки препинания в конец каждой строки, в обратном порядке. Использовать стек.
12. Из заданного текста перенести все строчные буквы в конец каждой строки, сохранив их порядок. Использовать очереди.
13. Из заданного текста перенести все строчные буквы в конец каждой строки, в обратном порядке. Использовать стек.
14. Из заданного текста перенести все прописные буквы в конец каждой строки, сохранив их порядок. Использовать очереди.
15. Из заданного текста перенести все прописные буквы в конец каждой строки, в обратном порядке. Использовать стек.
16. Из заданного текста перенести все пробелы в конец каждой строки. Использовать стек.
17. Создайте алгоритм сложения двух больших целых чисел, заданных в виде набора цифр хранимых с помощью очереди. Числа вводятся в виде строк с дальнейшим разбиением на цифры и помещением этих цифр в очередь
18. Используя очередь решить задачу: в файле записан текст, сбалансированный по круглым скобкам. Требуется для каждой пары соответствующих открывающейся и закрывающейся скобок напечатать номера их позиций в тексте, упорядочив пары номеров по возрастанию номеров позиций закрывающих скобок

Пример: для текста a+(45-f(x)(b-c)) надо напечатать: 8 10; 12 16; 3 17;*

19. Используя очередь решить задачу: в файле записан текст, сбалансированный по круглым скобкам. Требуется для каждой пары соответствующих открывающейся и закрывающейся скобок напечатать номера их позиций в тексте, упорядочив пары номеров по возрастанию номеров позиций открывающих скобок

Пример: для текста a+(45-f(x)(b-c)) надо напечатать: 3 17; 8 10; 12 16;*

20. Задача «Поступление товаров на склад». Описать структурный тип для группы пришедших на склад товаров с полями: количество единиц и закупочная цена единицы товара. При приеме товара в очередь добавляется группа однотипных товаров. При продаже начиная с начала очереди списывается введенное количество единиц по цене продажи (цена продажи не ниже закупочной). Группа товаров удаляется из очереди, если все единицы группы были проданы. Нужно считать количество товара в очереди и общую стоимость этих товаров, а также прибыль от продажи. Прибыль от продажи = Количество проданных единиц товара*(закупочная цена – цена продажи). Реализовать меню для работы с очередью с пунктами: Поступление товара; Продажа товара; Отчет (количество единиц на складе, стоимость товара на складе, прибыль от продажи).

Требования к оформлению программ:

1. **Содержание**. Программа должна делать то, что предусмотрено заданием, и больше ничего делать не должна
2. **Спецификация.** В преамбуле программы в комментариях указывать сведения:
   - Кто выполнил.
   - Что делает программа (кратко).
   - Что на входе, имена входных файлов указываются.
   - Что на выходе (что является результатов работы программы).
3. **Ввод и вывод**
   - Приглашение пользователю. (Например, сколько чисел, какого типа и через какой разделитель нужно вводить).

- Контрольный вывод (все введенные данные выводить на экран, и только после этого выполнять необходимые вычисления.)
- «Защита от дурака». Проверять вводимые данные на корректность. Например, если необходимо считать количество чего – то, то эта величина не может быть отрицательной и т.д.

4. **Структура кода.** Набираемый код должен быть хорошо структурированным. Использовать:
   - Отступы.
   - Комментарии – поясняют решение программы.
   - Осмысленные названия переменных.
   - Пояснения о назначении переменных в комментариях (кроме счетчиков).
5. **Декомпозиция кода**
   - Функциональная. Программу оформлять с помощью функций.
6. **Многофайловые проекты**
   - Все проекты должны состоять минимум из двух модулей: главного и подключаемого.

   Главный модуль **main.cpp**. В нем оставить только функцию main, в которой вызывать функции, описанные в других модулях.
   Модуль с описанием пользовательских функций **file.cpp** и **file.h.**В этот модуль перенести определение всех функций,необходимых для выполнения задания.

Контрольные вопросы:

1. Что такое стек?
2. Что такое очередь?
3. Что означает правило LIFO? Для какой динамической структуры данных это правило характерно?
4. Что означает правило FIFO? Для какой динамической структуры данных это правило характерно?
5. Что такое вершина стека?
6. Что такое список?
7. Что такое связный список?
8. Что такое несвязный список?
9. Что такое односвязный список?
10. Что такое двусвязный список?