Практическое занятие по дисциплине «Алгоритмы и структуры данных»

Вариант выбирается по **двум последним** цифрам номера студенческого билета **AB** (не совпадает с вариантом к контрольной работе).

Хранящиеся в списке данные выбираются по сумме цифр **A** и **B** (см. таблицу), если **A**+**B** = 0, то выбирается вариант **10**:

1	Беззнаковые символы (**unsigned char**)	10	Действительные числа (**double**)
2	Знаковые символы (**signed char**)	11	Действительные числа (**long float**)
3	Берзнаковые короткие целые числа (**unsigned short**)	12	Действительные числа (**long double**)
4	Знаковые короткие целые числа (**signed short**)	13	Символы (**char**)
5	Беззнаковые целые числа (**unsigned int**)	14	Целые числа (**int**)
6	Знаковые целые числа (**signed int**)	15	Короткие целые числа (**short**)
7	Беззнаковые длинные целые числа (**unsigned long**)	16	Длинные целые числа (**long**)
8	Знаковые длинные целые числа (**signed long**)	17	Действительные числа (**float**)
9	Действительные числа (**float**)	18	Действительные числа (**double**)

Программа работы со связным списком

1. Создать тип данных, описывающий звено двусвязного списка.
2. Создать процедуры работы с кольцевым двусвязным списком.
3. Создать ведущее звено кольцевого двусвязного списка.
4. Заполнить список данными, используя, например, цикл ***for*** и добавляя данные в начало списка (за ведущим звеном). Число данных выбрать в количестве 7 – 8 элементов.
5. Просмотреть содержимое списка.
6. Удалить звено, следующее за ведущим.
7. Просмотреть содержимое списка.
8. Удалить звено из середины списка, используя операцию поиска данных в списке.
9. Просмотреть содержимое списка.

Работа со списком как со стеком

10. Добавить 1 звено в начало списка (в позицию за ведущим звеном).
11. Просмотреть содержимое списка.
12. Считать данные из вершины стека (т.е. из начала списка – из звена, следующего за ведущим).
13. Просмотреть содержимое списка.
14. Повторить пункты 10, 11 несколько раз с разными данными.
15. Повторить пункты 12, 13 несколько раз.
16. Считать все звенья из начала списка. После каждой операции считывания выполнять просмотр списка.

Работа со списком как с очередью

17. Добавить 1 звено в конец очереди (т.е. в конец списка – в позицию перед ведущим звеном).
18. Просмотреть содержимое списка.
19. Повторить пункты 17, 18 несколько раз с разными данными.
20. Считать данные из начала очереди (из звена следующего за ведущим звеном).
21. Просмотреть содержимое списка.
22. Повторить пункты 20, 21 несколько раз.
23. Считать все звенья из начала списка. После каждой операции считывания выполнять просмотр списка.

Работа со списком как с двухвходовой очередью (деком)

24. Добавить 1 звено в конец очереди (т.е. в конец списка – в позицию перед ведущим звеном).
25. Просмотреть содержимое списка.
26. Повторить пункты 24, 25 несколько раз.
27. Удалить звено из начала очереди (из звена следующего за ведущим звеном).
28. Просмотреть содержимое списка.
29. Добавить 1 звено в начало очереди (в позицию за ведущим звеном).
30. Просмотреть содержимое списка.
31. Повторить пункты 29, 30 несколько раз.
32. Удалить звено из конца очереди.
33. Просмотреть содержимое списка.
34. Удалить все звенья из списка.

Пояснения к выполнению задания

Для каждой группы заданий можно создать отдельную программу, либо все пункты выполнять в единственной программе.

Основные процедуры, реализующие операции со связными списками, рассмотрены в конспекте лекций. Отдельные процедуры, например, просмотра линейного списка, требуют модификации для случая кольцевого списка. Такую модификацию студентам предлагается выполнить самостоятельно, например, путём сравнения и анализа операций поиска в линейных и кольцевых списках.

Для выбора операций со всеми исследуемыми структурами данных можно (но не обязательно) использовать меню, пример реализации которого находится по адресу http://saod.net/program.txt . При другом варианте выполнения задания пункты выполняются последовательно, в единой программе или в отдельной программе для каждой группы заданий.

При работе со стеком, очередью и деком заносить и извлекать данные можно теми же операциями, что и для обычного связного списка. Следует только учесть, что операция считывания должна быть «разрушающей», т.е. считанная информация должна удаляться из структуры данных. В случае использования связного списка для организации стека, очереди или дека это происходит буквально. Тогда разрушающая операция считывания может реализовываться следующим образом:

```
int Read(Zveno2* link)
{
  int data = link->next->a;
  Delete(link->next);
  return data;
}
```

Здесь **Zveno2** – это тип звена списка (двусвязного кольцевого), **a** – данные, хранящиеся в каждом звене списка, типа **int** (по варианту), **Delete** – функция удаления звена из двусвязного кольцевого списка. Одна и та же функция может использоваться для считывания данных из стека, очереди и начала дека. Для считывания данных из конца дека можно использовать похожую функцию:

```
int DequeRead(Zveno2* link)
{
  int a = link->prev->a;
  Delete(link->prev);
  return a;
}
```

Для занесения данных в конец очереди (т.е. в стандартную для очереди позицию для ввода данных) также можно использовать отдельную функцию:

```
void QueueWrite(Zveno2* link, int data)
{
  Insert(link->prev, data);
}
```

Здесь **Insert** – функция добавления звена в двусвязный кольцевой список.

Использоваться эти функции могут следующим образом:

Zveno2 *Head; – указатель на ведущее звено списка (операции создания ведущего звена не показаны).

Insert(Head, 1); – ввод данных в позицию за ведущим звеном списка (она же – вершина стека, она же – начало дека).

QueueWrite(Head, 2); – запись данных в конец очереди или дека.

Insert(Head->prev, 3); – другой вариант этой же операции, без использования отдельной функции.

cout << Read(Head) << … ; – считывание данных из вершины стека, начала очереди или дека.

cout << DequeRead(Head) << … ; – считывание данных из конца дека.

Использование специальных функции не обязательно, их можно заменить операторами, находящимися внутри функций. Выбор окончательного варианта оставляется на усмотрение студента.

При использовании символьных типов данных следует выбрать в качестве заносимых в список или дерево данных символьные переменные или константы, например:

Insert(Head, ’1’); или Insert(Head, ’a’);

Разрешается использовать технологию объектно-ориентированного программирования. В этом случае тип звена списка будет классом, а все необходимые функции – его функциями-членами.