

Класичний приватний університет

Кафедра програмування та інформаційних технологій

Борю С.Ю

МЕТОДИЧНІ ВКАЗІВКИ
ДО ВИКОНАННЯ КОНТРОЛЬНОЇ РОБОТИ З ДИСЦИПЛІНИ

Об'єктно-орієнтоване програмування *Друга частина*

для студентів денного і заочного відділення спеціальності
"Програмне забезпечення автоматизованих систем" та
„Системний аналіз та управління”

Запоріжжя, 2007

Методические указания по выполнению контрольной работы по курсу «об`єктно - орієнтоване програмування» для студентів денного і заочного відділення спеціальності "Програмне забезпечення автоматизованих систем" та „Системний аналіз та управління” /сост. Борю С.Ю. - Запорожье: Класичний приватний університет, 2007 г. – с.

Данные методические указания включают примеры расчетов, варианты контрольной работы и рекомендуемые источники, которые можно использовать при выполнении работы.

Целью контрольной работы является закрепление и углубление знаний, полученных на лекциях, практических занятиях и в процессе самостоятельного изучения материала по курсу «об`єктно - орієнтоване програмування».

Контрольная работа предназначена для проверки теоретических и практических знаний студентов; для отработки навыков самостоятельного изучения предмета, работы с литературой и программным обеспечением, умение логично и последовательно излагать усвоенный материал.

Содержание

ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ. .	4
ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА ПО КОНТРОЛЬНОЙ РАБОТЕ.	4
ЗАДАНИЕ 1 - "КЛАССЫ И ОБЪЕКТЫ"	5
ЗАДАНИЕ 2 «ДИНАМИЧЕСКИЕ СТРУКТУРЫ»	10
СПИСОК РЕКОМЕНДУЕМЫХ ИСТОЧНИКОВ	14
ПРИЛОЖЕНИЕ 1.....	15
ПРИЛОЖЕНИЕ 2.....	16
ПРИЛОЖЕНИЕ 3.....	29

Общие требования к выполнению контрольной работы.

Для выполнения контрольной работы необходимо:

- 1) изучить теоретический материал;
- 2) ознакомиться с приведенным примером;
- 3) выбрать вариант по следующей формуле – номер варианта = остаток от деления номера Вашей зачечной книжки на число 24.
- 4) выполнить работу;
- 5) оформить отчет по работе, используя текстовый процессор MS Word;
- 6) защита контрольной работы предполагает наличие электронного варианта.

Требования к оформлению отчета по контрольной работе.

1. Содержание работы оформить как автоматическое оглавление с указанием страниц и заполнителя.
2. Предусмотреть обычные сноски в тексте на использованные источники (литература, программное обеспечение).
3. Список источников оформить как нумерованный список.
4. В работе предусмотреть следующие разделы:
 - Титульный лист,
 - Содержание,
 - Теоретические сведения,
 - Порядок выполнения работы,
 - Список использованных источников.
5. Создать следующие колонтитулы для разделов:
 - верхний - слева Ваша ФИО, справа - название раздела;
 - нижний – слева номер варианта, справа - нумерация страниц;
 - титульный лист – колонтитулы отсутствуют.
6. Теоретические сведения оформляются в текстовом процессоре Word, с описанием типов и форматов данных, видов ссылок, формул, синтаксиса функций, построения диаграмм.
7. Описать порядок выполнения работы.

Задание 1 - "Классы и объекты"

Цель: Разработка простейших классов на примере разработки моделей элементарных объектов и динамических информационных структур (одно и двунаправленных списков).

Разработка простых классов

Постановка задачи

Разработать класс, набор методов (конструктор и минимум два метода) для программной модели заданного объекта. Описание объекта и его основных свойств приводится ниже. Разработать вызывающую программу (main), использующей объекты разработанного класса и тестирующие работоспособность всех методов.

Варианты заданий

1. Объект «комплексные числа». Операции определяются по обще принятым формулам. Предусмотреть возможность операции присваивания, сложения, умножения и перевода в текстовую строку текущих значений. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
2. Объект «комплексные числа». Операции определяются по обще принятым формулам. Предусмотреть возможность операции присваивания, вычитания, умножения и перевода в текстовую строку текущих значений. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
3. Объект «комплексные числа». Операции определяются по обще принятым формулам. Предусмотреть возможность операции присваивания, сложения, деления и перевода в текстовую строку текущих значений. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
4. Объект «комплексные числа». Операции определяются по обще принятым формулам. Предусмотреть возможность операции присваивания, сложения, умножения и перевода в показательную $(A \cdot e^{i\varphi})$ форму с возможностью распечатки на консоль.

Конструктор должен позволить создавать объекты без и с начальной инициализацией.

5. Объект «вектор на плоскости» заданный в системе декартовых координат. Начало вектора расположено в начале координат. Операции определяются согласно обще принятых формул линейной (векторной) алгебры. Предусмотреть возможность операции присваивания, сложения, скалярного умножения и распечатки координат текущих значений. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
6. Объект «вектор на плоскости» заданный в системе декартовых координат. Начало вектора расположено в начале координат. Операции определяются согласно обще принятых формул линейной (векторной) алгебры. Предусмотреть возможность операции присваивания, вычитания, скалярного умножения и распечатки координат текущих значений. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
7. Объект «вектор на плоскости» заданный в системе декартовых координат. Начало вектора расположено в начале координат. Операции определяются согласно обще принятых формул линейной (векторной) алгебры. Предусмотреть возможность операции присваивания, сравнения модулей, скалярного умножения и распечатки координат текущих значений. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
8. Объект «вектор на плоскости» заданный в системе декартовых координат. Начало вектора расположено в начале координат. Операции определяются согласно обще принятых формул линейной (векторной) алгебры. Предусмотреть возможность операции присваивания, нахождения угла между векторами, скалярного умножения и распечатки координат текущих значений. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
9. Объект «равнобедренный треугольник заданный длинами сторон». Предусмотреть возможность операции присваивания, определения площади и периметра, а так же логический метод, определяющий существует или такой треугольник. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
10. Объект «равносторонний треугольник заданный длинами сторон». Предусмотреть возможность операции присваивания, определения площади и периметра, а так же логический метод, определяющий существует или такой треугольник. Конструктор

должен позволить создавать объекты без и с начальной инициализацией.

11. Объект «прямоугольный треугольник заданный длинами сторон». Предусмотреть возможность операции присваивания, определения площади и периметра, а так же логический метод, определяющий существует или такой треугольник. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
12. Объект «равнобедренный треугольник заданный длиной равнобедренной стороной и углом между ними». Предусмотреть возможность операции присваивания, определения площади и периметра, а так же логический метод, отвечающий на вопрос – остро или тупо угольным является заданный треугольник. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
13. Объект «треугольник заданный длиной двух стороной и углом между ними». Предусмотреть возможность операции присваивания, определения площади и периметра, а так же логический метод, отвечающий на вопрос – остро или тупо угольным является заданный треугольник. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
14. Объект «прямоугольник заданный длинами двух сторон». Предусмотреть возможность операции присваивания, определения площади и периметра, а так же логический метод, отвечающий на вопрос – является ли прямоугольник квадратом. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
15. Объект «множество целых чисел заданной мощности». Предусмотреть возможность операции присваивания, объединения двух множеств, вывода на печать элементов множества, а так же метод отвечающий на вопрос – принадлежит ли указанное значение множеству. Конструктор должен позволить создавать объекты без и с начальной инициализацией. Мощность множества задается при создании объекта.
16. Объект «множество вещественных чисел заданной мощности». Предусмотреть возможность операции присваивания, объединения двух множеств, вывода на печать элементов множества, а так же метод отвечающий на вопрос – принадлежит ли указанное значение множеству. Конструктор должен позволить создавать объекты без и с начальной инициализацией. Мощность множества задается при создании объекта.

17. Объект «множество символов заданной мощности». Предусмотреть возможность операции присваивания, объединения двух множеств, вывода на печать элементов множества, а так же метод отвечающий на вопрос – принадлежит ли указанное значение множеству. Конструктор должен позволить создавать объекты без и с начальной инициализацией. Мощность множества задается при создании объекта.
18. Объект «множество целых чисел удвоенной длины заданной мощности». Предусмотреть возможность операции присваивания, объединения двух множеств, вывода на печать элементов множества, а так же метод отвечающий на вопрос – принадлежит ли указанное значение множеству. Конструктор должен позволить создавать объекты без и с начальной инициализацией. Мощность множества задается при создании объекта.
19. Объект «множество вещественных чисел удвоенной точности заданной мощности». Предусмотреть возможность операции присваивания, объединения двух множеств, вывода на печать элементов множества, а так же метод отвечающий на вопрос – принадлежит ли указанное значение множеству. Конструктор должен позволить создавать объекты без и с начальной инициализацией. Мощность множества задается при создании объекта.
20. Объект «множество байт заданной мощности». Предусмотреть возможность операции присваивания, объединения двух множеств, вывода на печать элементов множества, а так же метод отвечающий на вопрос – принадлежит ли указанное значение множеству. Конструктор должен позволить создавать объекты без и с начальной инициализацией. Мощность множества задается при создании объекта.
21. Объект «множество целых чисел не заданной (переменной) мощности». Предусмотреть возможность операции добавить элемент к множеству, определение количество элементов в множестве, вывода на печать всех элементов множества, а так же метод удаляющий указанный элемент из множества, если этот элемент принадлежит множеству. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
22. Объект «множество вещественных чисел не заданной (переменной) мощности». Предусмотреть возможность операции добавить элемент к множеству, определение количество элементов в множестве, вывода на печать всех элементов множества, а так же метод удаляющий указанный элемент из множества, если этот элемент принадлежит множеству.

Конструктор должен позволить создавать объекты без и с начальной инициализацией.

23. Объект «множество символов не заданной (переменной) мощности». Предусмотреть возможность операции добавить элемент к множеству, определение количество элементов в множестве, вывода на печать всех элементов множества, а так же метод удаляющий указанный элемент из множества, если этот элемент принадлежит множеству. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
24. Объект «множество целых чисел удвоенной длины не заданной (переменной) мощности». Предусмотреть возможность операции добавить элемент к множеству, определение количество элементов в множестве, вывода на печать всех элементов множества, а так же метод удаляющий указанный элемент из множества, если этот элемент принадлежит множеству. Конструктор должен позволить создавать объекты без и с начальной инициализацией.
25. Объект «прямоугольник заданный длинами двух сторон». Предусмотреть возможность операции присваивания, определения площади и периметра, а так же логический метод, отвечающий на вопрос – содержится ли, указанный параметрами метода прямоугольник , внутри прямоугольника. Конструктор должен позволить создавать объекты без и с начальной инициализацией

Примеры подобных учебных реализаций можно посмотреть в приложении 2

Задание 2 «динамические структуры»

«Информационные динамические структуры»

Постановка задачи

Написать программу, в которой создаются динамические структуры, и выполнить их обработку в соответствии со своим вариантом.

Для каждого вариант разработать следующие методы:

1. Конструктор пустого списка.
2. Добавление элемента в список (в соответствии со своим вариантом).
3. Удаление элемента из списка (в соответствии со своим вариантом).
4. Печать списка.
5. Запись списка в файл.
6. Восстановление списка из файла.
7. Деструктор списка (уничтожение).

Порядок выполнения работы

1. Разработать описание класса, выделить публичные и приватные поля данных. Разработать интерфейс класса – прототипы методов.
2. Написать функцию для создания списка. Функция может создавать пустой список, а затем добавлять в него элементы.
3. Написать функцию для печати списка. Функция должна предусматривать вывод сообщения, если список пустой.
4. Написать функции для удаления и добавления элементов списка в соответствии со своим вариантом.
5. Выполнить изменения в списке и печать списка после каждого изменения.
6. Написать функцию для записи списка в файл.
7. Написать функцию для уничтожения списка.
8. Записать список в файл, уничтожить его и выполнить печать (при печати должно быть выдано сообщение "Список пустой").
9. Написать функцию для восстановления списка из файла.
10. Восстановить список и распечатать его.
11. Уничтожить список.

Варианты заданий

1. Записи в линейном списке содержат поле данных звена типа int. Сформировать однонаправленный список. Удалить из него элемент с заданным номером, добавить элемент с заданным номером;
2. Записи в линейном списке содержат поле данных звена типа int. Сформировать однонаправленный список. Удалить из него элемент с заданным ключом, добавить элемент перед элементом с заданным ключом;
3. Записи в линейном списке содержат поле данных звена типа int. Сформировать однонаправленный список. Удалить из него К элементов, начиная с заданного номера, добавить элемент перед элементом с заданным ключом;
4. Записи в линейном списке содержат поле данных звена типа int. Сформировать однонаправленный список. Удалить из него элемент с заданным номером, добавить К элементов, начиная с заданного номера;
5. Записи в линейном списке содержат поле данных звена типа int. Сформировать однонаправленный список. Удалить из него К элементов, начиная с заданного номера, добавить К элементов, начиная с заданного номера;
6. Записи в линейном списке содержат поле данных звена типа int. Сформировать двунаправленный список. Удалить из него элемент с заданным номером, добавить элемент в начало списка.
7. Сформировать двунаправленный список. Удалить из него первый элемент, добавить элемент в конец списка.
8. Записи в линейном списке содержат поле данных звена типа int. Сформировать двунаправленный список. Удалить из него элемент после элемента с заданным номером, добавить К элементов в начало списка.
9. Записи в линейном списке содержат поле данных звена типа int. Сформировать двунаправленный список. Удалить из него К элементов перед элементом с заданным номером, добавить К элементов в конец списка.

10. Записи в линейном списке содержат поле данных звена типа `int`. Сформировать двунаправленный список. Добавить в него элемент с заданным номером, удалить `K` элементов из конца списка.
11. Записи в линейном списке содержат поле данных звена типа `*char`(строка символов). Сформировать двунаправленный список. Удалить из него элемент с заданным ключом, добавить элемент с указанным номером.
12. Записи в линейном списке содержат поле данных звена типа `*char`(строка символов). Сформировать двунаправленный список. Удалить из него Элементы, с одинаковыми ключевыми полями. Добавить элемент после элемента с заданным ключевым полем.
13. Записи в линейном списке содержат поле данных звена типа `*char`(строка символов). Сформировать двунаправленный список. Удалить из него `K` первых элементов. Добавить элемент после элемента, начинающегося с указанного символа.
14. Записи в линейном списке содержат поле данных звена типа `*char`(строка символов). Сформировать двунаправленный список. Удалить из него `K` элементов с указанными номерами. Добавить `K` элементов с указанными номерами.
15. Записи в линейном списке содержат поле данных звена типа `*char`(строка символов). Сформировать двунаправленный список. Удалить `K` элементов из конца списка. Добавить элемент после элемента с заданным ключом.
16. Записи в линейном списке содержат поле данных звена типа `*char`(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить `K` элементов в конец списка.
17. Записи в линейном списке содержат поле данных звена типа `*char`(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным номером. Добавить `K` элементов в начало списка.
18. Записи в линейном списке содержат поле данных звена типа `*char`(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить `K` элементов в начало списка.

19. Записи в линейном списке содержат поле данных звена типа *char(строка символов). Сформировать двунаправленный список. Удалить K элементов с заданными номерами. Добавить K элементов в начало списка.
20. Записи в линейном списке содержат поле данных звена типа *char(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить по K элементов в начало и в конец списка.
21. Записи в линейном списке содержат поле данных звена типа *char(строка символов). Сформировать двунаправленный список. Удалить элементы перед и после элемента с заданным ключом. Добавить по K элементов в начало и в конец списка.
22. Записи в линейном списке содержат поле данных звена типа *char(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить K элементов перед элементом с заданным ключом.
23. Записи в линейном списке содержат поле данных звена типа *char(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить K элементов после элемента с заданным ключом.
24. Записи в линейном списке содержат поле данных звена типа *char(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным номером. Добавить по K элементов перед и после элемента с заданным ключом.
25. Записи в линейном списке содержат поле данных звена типа *char(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить K элементов перед элементом с заданным номером.

Содержание отчета

1. Постановка задачи.
2. Описание класса, методы для работы со списком.
3. Функция main().
4. Результаты выполнения работы.

Примеры подобных учебных реализаций можно посмотреть в приложении 3

Список рекомендуемых источников

1. Подбельский В. В., Фомин С. С. Программирование на языке Си: Учеб. пособие. –М.:Финансы и статистика, 1998.–600с.
2. Подбельский В. В. Язык Си++: Учеб. пособие.–М.:Финансы и статистика, 1996.–560с.
3. Страуструп Б. Язык программирования Си++: Пер. с англ. – М.: Радио и связь, 1991.-352с.

Приложение 1
Класичний приватний університет

Кафедра програмування та інформаційних технологій

“Отримано”

Реєстраційний номер № _____
від “____” _____ 200__ р.

“Відправлено з зауваженнями”

Реєстраційний номер № _____
від “____” _____ 200__ р.

“Отримано повторно”

Реєстраційний номер № _____
від “____” _____ 200__ р.

КОНТРОЛЬНА РОБОТА № _____

з дисципліни _____
на тему _____

Виконав (ла)
студент (ка) _____ курсу, _____ групи _____
(прізвище, ім'я, по батькові)

Перевірів:

_____ (оцінка, дата, підпис викладача) _____ (прізвище, ім'я, по батькові викладача)

Адреса університету:

69002, м.Запоріжжя,
вул. Жуковського, 70-б.

тел. 63-99-73

Адреса для повідомлення
результату контрольної роботи
студента:

_____ (індекс, населений пункт, вулиця, будинок, квартира)

телефон: _____

м. Запоріжжя, 200__ р.

Приложение 2

// СТЕК Символьных строк

```
class stack
{
    private:
        char st[55];    // массив для хранения элементов стека
        int ptr;        // указатель номера СЕДУЩЕГО свободного места
        в массиве
        int max_ptr;    // максимальный размер стека
        char buf[110];  // для организации печати информации о
        состоянии стека
    public:
        stack();        // конструктор
        ~stack();       // деструктор
        void push(char c); // метод - поместить в стек
        char pop();      // метод - взять из стека ('\0' - если
        стек пуст)
        char look();     // метод - посмотреть что в вершин стека
        ('\0' - если стек пуст)
        char * pr();     // метод - вернуть символьную строку -
        элементов в стеке
        void prln();     // метод - распечатать стек
};

stack::stack()          // конструктор
{
    max_ptr=55;
    ptr=0;
    for(int i=0; i< max_ptr; i++) st[i]='\0';
};

stack::~~stack()        // деструктор
{
    ptr=0;
    for(int i=0; i< max_ptr; i++) st[i]='\0';
};

void stack::push(char c) // метод - поместить в стек
{
    if ( (ptr+1) > max_ptr)
    { printf("*** переполнение стека ***\n");
      exit(666);
    };
    st[ptr++]=c;
    return;
};

char stack::pop()       // метод - взять из стека ('\0' - если стек
пуст)
{
    char cc;
    if (ptr < 0 ) return '\0';
    ptr--;
```



```

        cc=st[ptr];
        st[ptr]='\0';
        return cc;
    };

char stack::look()          // метод - посмотреть что в вершин стека
(' \0' - если стек пуст)
{
    char cc;
    int t_ptr;
    t_ptr=ptr-1;
    if ( t_ptr < 0 ) return '\0';
    cc = st[t_ptr];
    return cc;
};

char *stack::pr()          // метод - вернуть символьную строку -
элементов в стеке
{
    int i;
    int j=0;
    buf[j++]=': '; buf[j++]=' ';
    for (i= (strlen(st)-1); i >= 0; i-- )
    {
        buf[j++] = st[i];
        buf[j++] = ' ';
    };
    buf[--j]='\0';
    //sprintf(buf, ": %s ", st);
    return buf;
};

void stack::prln()          // метод - распечатать стек
{
    printf("%s\n", pr() );
    return;
};

=====
===
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// #include "stack.h"
void main(int argc, char * argv[])
{
    stack st;
    st.prln();

    st.push('1'); st.prln();
    st.push('2'); st.prln();
    st.push('3'); st.prln();
    st.push('4'); st.prln();
    char
    z=st.pop();
    printf("==>pop ==> %c \n", z);
}

```

```

    st.prln();
    z=st.pop();
    printf("==>pop ==> %c \n",z);
    st.prln();
    z=st.look();
    printf("==>look==> %c \n",z);
    st.prln();
    printf("***** %s\n",st.pr());
    return;
};
C:\Test\cpp\bsu-stack>test.exe
:
: 1
: 2 1
: 3 2 1
: 4 3 2 1
==>pop ==> 4
: 3 2 1
==>pop ==> 3
: 2 1
==>look==> 2
: 2 1
***** : 2 1

```

Класс - "моя строка" st

```

#include<stdio.h>
#include <stdlib.h>
#include <string.h>

class st
{ private:
    int l;           // длина
    char * s;        // адрес строки
public:
    st();             // конструктор без параметра
    st(char *);       // конструктор с инициализирующим параметром
    ~st();
    const char * put(const char *); // взять указатель на
                                     строку
    const char * get();           // записать строку
    st(const st & x);             // конструктор
                                     копирования
    st & operator=(const st & x); // оператор
                                     присваивания
    st & operator+=(const st & x); //унарный оператор +=

    friend
    st operator+(const st & x, const st & y);
                                     // дружественная функция +

    friend

```

```

    st operator-(const st & x, const st & y);
                                   // дружественная функция -

    void ch(char); // для теста - изменить 0-й символ

    void pr( char *); // для теста - распечатать....

};

void st::ch(char x)
{ if (s[0] != '\0') this->s[0]=x;
  return;
};

    void st::pr(char tit[])=""
{printf("%s[%d]<%s>\n",tit,l,s);return; };

st::st()
{ l=1; s=new char[l]; s[0]='\0';};

st::st(char * d)
{ l=strlen(d)+1; s=new char[l];
  strcpy(s, d);
};

st::~~st()
{
    delete [] s;
    l=0;
};

st & st::operator=(const st & x)
{
    if( this == &x ) return *this;
    delete [] this->s;
    this->l=x.l;
    this->s=new char[l];
    strcpy(s, x.s);
    return *this;
};

// КОСТРУКТОР КОПИРОВАНИЯ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//    попробуйте убрать.....
st::st(const st & x)
{
    this->l=x.l;
    this->s=new char[l];
    strcpy(s, x.s);
};

// унарный +=
st & st::operator+=(const st & x)

```

```

{
    char *p;    int lp=this->l+x.l-1;
    p=new char[lp];
    sprintf(p,"%s%s",this->s, x.s);
    // в p - конкатенированная строка
    delete [] this->s;
    this->l=lp;
    this->s=new char[lp];
    strcpy(this->s, p);
    return *this;
};

// бинарный +
st operator+(const st & x, const st & y )
{
    char *p;
    st temp; //! вызывается конструктор копирования
    int lp=x.l + y.l -1;

    p=new char[lp];

    sprintf(p,"%s%s", x.s, y.s);
    // в p - конкатенированная строка
    temp.l=lp;
    temp.s=new char[lp];
    strcpy(temp.s, p);
    return temp;
};

// бинарный -
st operator-(const st & x, const st & y )
{
    printf("x=%s  y=%s \n", x.s, y.s);
    char *p;
    st temp;  // !!!!!!!!!!!!!!!!!!!!!!!
    int i,j;
    int k;
    char tchar;
    int flag;
    int n=x.l+1;
    p=new char[n];
    k=0;
    for(i=0; i< x.l; i++)
    {
        tchar=x.s[i];
        // есть ли tchar в y.s ?
        flag=0;
        for(j=0;j<y.l;j++)
            if(tchar==y.s[j]) {flag=1; break;};
        if(flag==0) p[k++]=tchar;
    };
    p[k]='\0';
    // в p - новая строка
    temp.l=k;
    temp.s=new char[k];

```

```

        strcpy(temp.s, p);
        return temp;
};

const char * st::put(const char * d)
{
    delete [] s;
    l=strlen(d)+1; s=new char[l];
    strcpy(s, d);
    return s;
};

const char * st::get()
{
    return s;
}

int main(int argc, char *argv[] )
{ st a1, a2, a3, a4="куку";
  const char * p;

  p=a1.put("привет");
  printf("***p==<%s>\n",p);
  a1.pr("a1= .... ==> ");
  a2=a1;
  a2.pr("a2=a1 ==> ");
  a3.put("Всем "); a3.pr("a3.put(...) ==> ");
  a3+=a2;          a3.pr("a3+=a2 ==> ");
  printf("*****\n");
  a1.ch('$');      a1.pr("a1($) ==> ");
  a2=a1;
  a2.ch('@');      a2.pr("a1(@) ==> ");
  a4.pr("a4 ==> ");
  printf("*****\n");
  a2=a1 + " ++ " + a4;    a2.pr("a1 ++ a4 ==> ");
  a1 = "победа!";    a1.pr("a1=.... ==> ");
  a2 = a1 - "по";    a2.pr("a2=a1 - по ==> ");
  a3 = a1 - "ап";    a3.pr("a3=a1 - ап ==> ");
  return 0;
}

=====
C:\Test\cpp\b_str>t
***p==<привет>
a1= .... ==> [7]<привет>
a2=a1 ==> [7]<привет>
a3.put(...) ==> [6]<Всем >
a3+=a2 ==> [12]<Всем привет>
*****
a1($) ==> [7]<$ривет>
a1(@) ==> [7]<@ривет>
a4 ==> [5]<куку>
*****
a1 ++ a4 ==> [16]<$ривет ++ куку>

```

```

a1=..... ==> [8]<победа!>
x=победа!   y=по
a2=a1 - по ==> [5]<беда!>
x=победа!   y=ап
a3=a1 - ап ==> [5]<обед!>

```

```

C:\Test\cpp\b_str>

```

МНОЖЕСТВА

```

#include <stdio.h>
#include <iostream.h>
#include <string.h>
#include <stdlib.h>
//множества
class set {
private:
int size;
int *elems;
public:
set(void);
friend set operator+(set, set);
friend set operator-(set, set);
friend set operator*(set, set);
friend int operator<=(set, set);
friend int operator>=(set, set);
friend int operator==(set, set);
friend int operator!=(set, set);
friend void operator>>(set, char*); //Вывод множества в файл
friend void operator<<(set&, char*); //Ввод множества из файла
void Print();
};

class ErrorInOpenFile {
public:
ErrorInOpenFile() { strcpy(mess, "\Ошибка при работе с файлом"); }
void ErrMessage() const { cout<<mess; }
private:
char mess[50];
};

set::set() // конструктор
{
size=0;
};

int operator != (set a, set b) // не равно
{
int i, j, flg, res;
res=1;

```

```

for(i=0; i<a.size; i++)
{
    flg=0;
    for(j=0; j<b.size; j++)
        if(b.elems[j] == a.elems[i]) flg=1;
    if(!flg) res=0;
}
return !res;
};

int operator == (set a, set b) // равно
{
    int i,j,flg,res;
    res=1;
    for(i=0;i<a.size;i++)
    {
        flg=0;
        for(j=0; j<b.size; j++)
            if(b.elems[j] == a.elems[i]) flg=1;
        if(!flg) res=0;
    }
    return res;
};

int operator >= (set a, set b)
{
    int i,j;
    int res,flg;
    res=1;
    for(i=0; i<a.size; i++)
    {
        flg=0;
        for(j=0; j<b.size; j++)
            if(b.elems[j] == a.elems[i]) flg=1;
        if(!flg) res=0;
    }
    return res;
};

int operator <= (set a, set b)
{
    int i,j;
    int res,flg;
    res=1;
    for(i=0; i<b.size; i++)
    {
        flg=0;
        for(j=0;j<a.size;j++)
            if(a.elems[j] == b.elems[i]) flg=1;
        if(!flg)res=0;
    }
    return res;
};

```

```

set operator + (set a, set b)
{
    int i, j, k;
    int count;
    count=0;
    int flag;
    for(i=0; i<b.size; i++)
    {
        flag=1;
        for(j=0; j<a.size; j++)
            if(a.elems[j] == b.elems[i]) flag=0;
        if(flag) count++;
    }
    count+=a.size;
    set res;
    res.size=count;
    res.elems=new int[count];
    for(i=0; i<a.size; i++) res.elems[i]=a.elems[i];
    for(j=0; j<b.size; j++)
    {
        flag=1;
        for(k=0; k<a.size; k++)
            if(a.elems[k]==b.elems[j]) flag=0;
        if(flag) res.elems[i++]=b.elems[j];
    }
    return res;
};

```

```

set operator-(set a, set b)
{
    int i, j, k;
    int count;
    count=0;
    int flag;
    for(i=0; i<a.size; i++)
    {
        flag=1;
        for(j=0; j<b.size; j++)
            if(a.elems[i]==b.elems[j]) flag=0;
        if(flag) count++;
    }
    set res;
    res.size=count;
    res.elems=new int[count];
    i=0;
    for(j=0; j<a.size; j++)
    {
        flag=1;
        for(k=0; k<b.size; k++)
            if(a.elems[j]==b.elems[k]) flag=0;
        if(flag) res.elems[i++]=a.elems[j];
    }
}

```



```

return res;
};

set operator*(set a,set b)
{
int i,j,k;
int count;
count=0;
int flag;
for(i=0;i<b.size;i++)
{
flag=1;
for(j=0;j<a.size;j++)
if(a.elems[j]==b.elems[i]) flag=0;
if(!flag) count++;
}
set res;
res.size=count;
res.elems=new int[count];
i=0;
for(j=0;j<b.size;j++)
{
flag=1;
for(k=0;k<a.size;k++)
if(a.elems[k]==b.elems[j]) flag=0;
if(!flag) res.elems[i++]=b.elems[j];
}
return res;
};

void operator<<(set& nw,char* FileName)
{
const MAX_STR=255;
FILE *in;
char buff[MAX_STR+1];
try
{
in=fopen(FileName,"r");
if(in==NULL) throw ErrorInOpenFile();
}
catch(ErrorInOpenFile ERR)
{
ERR.ErrMessage();
exit(1);
}
unsigned i,count;
fgets(buff,MAX_STR,in);
sscanf(buff,"%d",&count);
nw.size=count;
nw.elems=new int[count];
for(i=0;i<count;i++)

```

```

{
fgets(buff,MAX_STR,in);
sscanf(buff,"%d",&nw.elems[i]);
}
fclose(in);
};

void operator>>(set nw,char* FileName)
{
int i;
const MAX_STR=255;
char buff[MAX_STR];
FILE* out;
try
{
out=fopen(FileName,"w+");
if(out==NULL)throw ErrorInOpenFile();
}
catch(ErrorInOpenFile ERR)
{
ERR.ErrMessage();
exit(1);
}
sprintf(buff,"%d\n",nw.size);
fputs(buff,out);
for(i=0;i<nw.size;i++)
{
sprintf(buff,"%d\n",nw.elems[i]);
fputs(buff,out);
}
fclose(out);
};

void set::Print()
{
int i;
cout<<"\n";
for(i=0;i<size;i++)
cout<<elems[i]<<"\n";
};

void main()
{
set A;
set B;
set C;
A<<"SetA.txt";
cout<<"\nSet A";
A.Print();
B<<"SetB.txt";

cout<<"\nSet B";
B.Print();

```

```

C=A+B;
cout<<"\nSet C";
C.Print();
// set D;
// D<<"SetD.txt";
// set E;
// E=A-D;
// cout<<"\nSet E";
// E.Print();
// set F;
// F=A*D;
// cout<<"\nSet F";
// F.Print();
};

```

Атрибуты строки

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

```

#include <conio.h>

```

```

class string

```

```

{
    private:
        char *str;
        unsigned char attr;
        int row, col;
    public:
        string();
        string(char *, unsigned char, int , int );
        void write();
};

```

```

string::string()

```

```

{
    str = new char [ sizeof "Приветик!!!!" ];
    strcpy(str, "Приветик!!!!");
    attr=(BLUE << 4) + YELLOW; // BROWN<<4+GREEN;
    row=15;
    col=36;
};

```

```

string::string(char * line, unsigned char a, int y=0, int x=0)

```

```

{
    str=new char [ strlen(line)+1 ];
    strcpy(str, line);
    attr=a;
    row=y;
    col=x;
};

```

```

void string::write()
{
    textattr( attr);
    gotoxy(col, row);
    cputs( str );
};

void main(int argc, char * argv[])
{
    clrscr();

    string string1;
    string string2("Сторка вторая!", (BLACK<<4)+WHITE);
    string string3("Третья сторка!", (BROWN<<4)+GREEN, 17, 19);
    string string4("Четвертая сторка!", (BROWN<<4)+GREEN, 1, 1);
    string1.write();
    string2.write();
    string3.write();
    string4.write();

    return;
};

```

Приложение 3

учебная реализаций (не полная) дву связанного списка (struct sp)

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

class spis
{
public:
    spis();
    int add_ends(const char *);    // добавим звено в конец списка
    int add_first(const char *);  // добавим звено в начало списка
    int set_tek(int n);           // устанавливает звено с номером n текущим звеном
                                   // n=0 - первое звено становится текущим
                                   // n=-1 - последнее звено становится текущим
                                   // возвращает 1 - если установка выполнена и
                                   // 0 - установить не удалось
                                   // (тек. звено не изменяется)
    char * get_tek(char * data);  // возвращает данные текущего звена
    char * put_tek(char * data);  // изменяет данные текущего звена

    int del_ends(); // удалить звено с конца списка
    int del_first(); // удалить звено с начала списка
    int del_tek();   // удаляет текущее звено
    int print(char * kod); // распечатать список "DOWN" в прямом "UP" - в обратном
    int print(char * kod, int n); // распечатать список "DOWN" в прямом "UP" - в обратном
                                   // с n звена от начала или конца списка с нумерацией звеньев
    int num(); // возвращает количество звеньев в списке
};
```

```

int add_tek_down(char * data); // вставка звена после текущего (sp_tek)
int add_tek_up(char * data);   // вставка звена перед текущим   (sp_tek)

~spis();

private:
    struct sp                // структура звена
    { sp * a_up;              // адрес предыдущего
      sp * a_down;            // адрес следующего
      char info[81];          // поле для хранения данных
    };

    sp *sp_first, *sp_ends, *sp_tek; // указатели на характерные звенья
//     первое      последнее      текущее

    void s_strcpy(char *, const char *in);

}; // end class spis

//=====
void spis::s_strcpy(char * outt, const char *in) // копирование данных
{
    int i=0;

    while ( (outt[i]=in[i]) != '\0')
    { i++;
      if ( i >=80 ) {outt[i]='\0'; break;};
    };

    return;
};
//=====

```

```

spis::spis() // инициализация списка
{ sp_first=sp_ends=sp_tek=NULL; //return 1;
};
//=====
spis::~spis() // удалить список из памяти
{ sp *a_tek, *a_next, *p;
  a_tek=sp_ends; // в a_tek - адрес последнего звена
  if (a_tek != NULL )
  {
    do
    {
      a_next=a_tek->a_up; // адрес предыдущего звена
      delete a_tek; // удалить текущее
      a_tek=a_next; // адрес "нового" последнего звена
    }
    while ( a_tek !=NULL);
  };
  sp_first=sp_ends=sp_tek=NULL;
};
//=====
int spis::add_ends(const char * data) // добавим в конец
{ sp *a_tek, *p, *a_next;
  if ( (sp_first == NULL) && (sp_ends == NULL) ) // нет звеньев ВООБЩЕ...
  {
    p = new sp; // адрес нового звена
    p->a_up=NULL; // в новом звене указатели вверх и вниз - пустые
    p->a_down=NULL;
    p->info[0]='\0'; // в новом звене инф. строка пустая
    strcpy(p->info, data); // запись из буфера в инф. строку списка
    sp_first=p; //голова списка
    sp_ends=p;
    sp_tek=p;
    return 1;
  };
  if ( sp_ends == NULL )

```

```

    { printf("*** ошибка структуры списка (sp_ends=null)\n"); exit(666); return 0;};

    a_tek=sp_ends; //- адрес последнего звена
    p = new sp; // адрес нового звена
    //printf("tek=<%p> new=<%p>\n", a_tek, p);
    a_tek->a_down = p; //ссылка вниз в предпоследнем звене
    p->a_up= a_tek; //ссылка вверх в последнем (новом) звене
    p->a_down=NULL; //ссылка вниз в последнем (новом) звене
    s_strcpy(p->info, data);
    sp_ends=p;
    sp_tek=p;
    return 1;
};
//=====
int spis::add_first(const char * data) // добавим в начало
{ sp *a_tek, *p, *a_next;
  if ( (sp_first == NULL) && (sp_ends == NULL) ) // нет звеньев ВОООЩЕ...
  {
    p = new sp; // адрес нового звена
    p->a_up=NULL; // в новом звене указатели вверх и вниз - пустые
    p->a_down=NULL;
    p->info[0]='\0'; // в новом звене инф. строка пустая
    s_strcpy(p->info, data); // запись из буфера в инф. строку списка
    sp_first=p; //голова списка
    sp_ends=p;
    sp_tek=p;
    return 1;
  };
  if ( sp_first == NULL )
  { printf("*** ошибка структуры списка (sp_first=null)\n"); exit(666); return 0;};
  a_tek=sp_first; // - адрес первого звена
  p = new sp; // адрес нового звена
  //printf("tek=<%p> new=<%p>\n", a_tek, p);
  p->a_up=NULL; // ссылка вверх в новом звене- нет
  p->a_down=a_tek; // ссылка вниз в новом звене - на бывшее первое звено

```



```

    s_strcpy(p->info, data);    // запись из буфера в инф. строку списка
    a_tek->a_up=p;              // ссылка вверх в бывшем первом звене на новое звено
    sp_first=p;                // новый указатель на "голову" звена
    sp_tek=p;
    sp_tek=p;
    return 1;
};
//=====
int spis::del_ends() // удалить запись с конца списка
{
    sp *a_tek, *p;
    a_tek=sp_ends;    // удаляемое звено...
    if ( a_tek == sp_first ) // удаляется первое звено - подправим указатель
        sp_first=NULL;
    if ( a_tek == NULL ) return NULL;
    p=a_tek->a_up;      // предыдущее звено
    if ( p != NULL )    // есть выше стоящее звено...
        { p->a_down=NULL; };
    sp_ends=p;
    sp_tek=p;
    delete a_tek;
    return 1;
};
//=====
int spis::del_first() // удалить запись с начала списка
{
    sp *a_tek, *p;
    a_tek=sp_first;    // удаляемое звено...
    if ( a_tek == sp_ends ) // удаляется последнее звено - подправим указатель
        sp_ends=NULL;
    if ( a_tek == NULL ) return NULL;
    p=a_tek->a_down;    // следующее звено
    if ( p != NULL ) // есть ниже стоящее звено...
        { p->a_up=NULL; };
    sp_first=p;

```

```

    sp_tek=p;
    delete a_tek;
    return 1;
};
//=====
int spis::print(char * kod) // распечатать список "DOWN" в прямом "UP" - в обратном
{
    sp *a_tek, *a_next, *p;
    if ( strcmp(kod, "DOWN")==0 )
    {
        // распечатка списка в прямом направлении .....
        //clrscr();
        a_tek= sp_first;
        printf("first=<%p> tek=<%p> ends=<%p>\n\n",sp_first, sp_tek, sp_ends);
        p = a_tek;
        while ( p != NULL )
        {
            printf("ADD=<%p> ZVENO=<%p><%p><%s>\n",p, p->a_up,p->a_down,p->info);
            //getchar();
            a_tek= p;
            p = a_tek->a_down;
        };
        printf("***** EOSisos (ends) ***** нажмите Enter\n");getchar();
        return 1;
    };
    if ( strcmp(kod, "UP")==0 )
    {
        // распечатка списка в обратном направлении
        a_tek=sp_ends; // - адрес последнего звена
        printf("first=<%p> tek=<%p> ends=<%p>\n\n",sp_first, sp_tek, sp_ends);
        p=a_tek; // адрес "вверх"
        while ( p != NULL )
        {
            printf("ADD=<%p> ZVENO=<%p><%p><%s>\n",p, p->a_up,p->a_down,p->info);
            //getchar();

```

```

        a_tek= p;
        p = a_tek->a_up;
    };
    printf("***** EOSisoc (first) ***** нажмите Enter\n");getchar();
    return 1;
};
return 0;
};
//=====
int spis::print(char * kod, int n) // распечатать список "DOWN" в прямом "UP" - в обратном
                                   // с n звена от начала или конца списка с нумерацией звеньев
{
    int num;
    sp *a_tek, *a_next, *p;
    if ( strcmp(kod, "DOWN")==0 )
    {
        // распечатка списка в прямом направлении .....
        //clrscr();
        a_tek= sp_first;
        // printf("first=<%p> tek=<%p> ends==<%p>\n\n",sp_first, sp_tek, sp_ends);
        p = a_tek;
        num=1;
        while ( p != NULL )
        {
            if ( num >= n)
                //printf("ADD=<%p> ZVENO=<%p><%p><%s>\n",p, p->a_up,p->a_down,p->info);
                printf("num=%04d zap=<%s>\n",num,p->info);
            num++;
            //getchar();
            a_tek= p;
            p = a_tek->a_down;
        };
        printf("***** EOS (ends) ***** нажмите Enter\n");getchar();
        return 1;
    };
    if ( strcmp(kod, "UP")==0 )

```

```

{
    // распечатка списка в обратном направлении
    a_tek=sp_ends;    // - адрес последнего звена
    printf("first=<%p> tek=<%p> ends==<%p>\n\n",sp_first, sp_tek, sp_ends);
    p=a_tek;    // адрес "вверх"
    num=1;
    while ( p != NULL )
    {
        if ( num >= n)
            //printf("ADD=<%p> ZVENO=<%p><%p><%s>\n",p, p->a_up,p->a_down,p->info);
            printf("num=%04d zap=<%s>\n",num,p->info);
            num++;
            //getchar();
            a_tek= p;
            p = a_tek->a_up;
        };
        printf("***** EOS (first) ***** нажмите Enter\n");getchar();
        return 1;
    };
    return 0;
};
//=====
int spis::num() // возвращает количество звеньев в списке
{ sp *a_tek, *p;
  int num=0;
  a_tek= sp_first;
  p      = a_tek;
  while ( p != NULL )
  {
      num++;
      //getchar();
      a_tek= p;
      p = a_tek->a_down;
  };
  return num;
}

```

```

};
//=====
int spis::add_tek_down(char * data) // вставка звена после текущего (sp_tek)
{
    sp *a_tek, *a_next, *a_new;
    if (sp_tek == sp_ends) return add_ends(data);
    a_tek=sp_tek;
    if (a_tek == NULL) return add_ends(data);
    a_next=a_tek->a_down;
    if (a_next == NULL) return add_ends(data);

    a_new = new sp;
    a_new->a_down=a_next;
    a_new->a_up=a_tek;
    s_strcpy(a_new->info, data);

    a_tek->a_down=a_new;
    a_next->a_up=a_new;

    sp_tek=a_new;

    return 1;
};
int spis::add_tek_up(char * data) // вставка звена перед текущим (sp_tek)
{
    sp *a_tek, *a_first, *a_new;
    if (sp_tek == sp_first) return add_first(data);

    a_tek=sp_tek;
    if (a_tek == NULL) return add_first(data);
    a_first=a_tek->a_up;
    if (a_first == NULL) return add_first(data);

    a_new = new sp;
    a_new->a_up=a_first;

```

```

    a_new->a_down=a_tek;
    s_strcpy(a_new->info, data);

    a_tek->a_up=a_new;
    a_first->a_down=a_new;

    sp_tek=a_new;

    return 1;
};
//=====
char * spis::get_tek(char * data) // возвращает данные текущего звена
{
    if ( sp_tek == NULL ) return NULL;
    s_strcpy(data, sp_tek->info);
    return ( data);

};
//=====
char * spis::put_tek(char * data) // изменяет данные текущего звена
{
    if ( sp_tek == NULL ) return NULL;
    s_strcpy(sp_tek->info, data);
    return ( data);

};
//=====
int spis::set_tek(int n) // устанавливает звено с номером n текущим звеном
                        // n=0 - первое звено становится текущим
                        // n=-1 - последнее звено становится текущим
                        // возвращает 1 - если установка выполнена и
                        // 0 - установить не удалось (тек. звено не изменяется)
{ int n_zven=num();
  sp *a_tek, *p;
  int num=0;

```

```

if (n == 0) {sp_tek = sp_first; return 1;};
if (n == -1) {sp_tek = sp_ends; return 1;};
if ( 1 <= n && n <= n_zven )
{
    a_tek= sp_first;
    p      = a_tek;
    while ( p != NULL )
    {
        num++;
        if ( num == n )
        {
            sp_tek=p;
            return 1;
        };
        if ( num > n ) break;
        a_tek= p;
        p = a_tek->a_down;
    };
    return 0;
};
return 0;
};
//=====
//sp_set_tek(char * data) // устанавливает звено содержащее data текущим
//sp_get_first(char * data) // читает список от начала к концу, начитая с текущего
// следующее звено после чтения становится текущим
//=====
//=====
//=====
//=====
//=====
int main(int argv, char * * argc)
{
    int k_zap=0, k;

```

```

    char z;
    char nfile[80], buf[80];
    FILE *h;
    //sp *first, *a_tek, *p, *a_next;
    if (argv > 1 ) strcpy(nfile,argv[1]);
    else { printf("Не указано имя файла для чтения.....\n"); return -2;};
//=====
    h=fopen(nfile, "r");
    if (h == NULL )
        { printf( "Cannot open input file <%s> \n", nfile); return -1; };
//=====

// инициализация списка.....
    spis SP;

next:
    if ( fgets(buf, 80, h) == NULL )
        { printf( "Конец файла <%s> \n", nfile);
          goto endd;
        };
    k=strlen(buf)-1; if (k>0) buf[k]='\0'; else goto endd;

    k_zap++;
    printf("прочитали %d <%s>\n", k_zap, buf);
// поместить новую запись из buf в список
    SP.add_ends(buf);
    goto next;
endd:
    fclose(h);
    getchar();

// распечатка списка в прямом направлении .....
    SP.print("DOWN");
// распечатка списка в обратном направлении

```



```

    SP.print("UP");

// добавим в конец....
    SP.add_ends("Добавка в конец аaaaaaa 111*****");
    SP.add_ends("Добавка в конец аaaaaaa 222*****");
    SP.add_ends("Добавка в конец аaaaaaa 333*****");

    printf("распечатка списка в прямом направлении .....\\n");
// распечатка списка в прямом направлении .....
    SP.print("DOWN");

// добавим в начало

    SP.add_first("Добавка в начало аaaaaaa 111*****");
    SP.add_first("Добавка в начало аaaaaaa 222*****");
    SP.add_first("Добавка в начало аaaaaaa 333*****");

// распечатка списка в прямом направлении .....
    SP.print("DOWN",1);

printf("В списке  %d звеньев\\n", SP.num()); getchar();

int nnn, sss;
char bbb[80];
do
{
    printf("укажи номер звена для распечатки или -5 "); scanf("%d", &nnn);
    sss= SP.set_tek(nnn);
    SP.get_tek(buf);
    printf("sss=%d тек. звено buf=<%s>\\n", sss, buf);
}
while ( nnn != - 5);

```

```

do
{   printf("укажи номер звена после которого вставить или -5 и текст звена\n");
    scanf("%d %s", &nnn, buf);

    sss= SP.set_tek(nnn);

    if ( sss==0)
        { printf("номер звена не установлен.....\n");
          continue;
        };
    printf("вставка после <%s>\n",SP.get_tek(bbb) );
    SP.add_tek_down(buf);
    SP.print("DOWN",1);
}
while ( nnn != - 5);

//return 1;
// goto ddd;

// распечатка списка в прямом направлении
SP.print("DOWN",1);

do
{   printf("укажи номер звена перед которым вставить или -5 и текст звена\n");
    scanf("%d %s", &nnn, buf);

    sss= SP.set_tek(nnn);

    if ( sss==0)
        { printf("номер звена не установлен.....\n");
          continue;
        };
    printf("вставка перед <%s>\n",SP.get_tek(bbb) );
    SP.add_tek_up(buf);

```

```

        SP.print("DOWN",1);

    }
while ( nnn != - 5);
// распечатка списка в прямом направлении
    SP.print("DOWN",1);
do
{   printf("укажи номер звена которое изменить или -5 и текст звена замены\n");
    scanf("%d %s", &nnn, buf);

    sss= SP.set_tek(nnn);

    if ( sss==0)
        {   printf("номер звена не установлен.....\n");
            continue;
        };
    printf("замена звена <%s>\n",SP.get_tek(bbb) );
    SP.put_tek(buf);
    SP.print("DOWN",1);

}
while ( nnn != - 5);
while ( SP.num() !=0 )
{
    printf("Удаляем запись СВЕРХУ\n" );
    printf("В списке до удаления %d звеньев\n", SP.num()); getchar();
    SP.del_first();
    printf("В списке ПОСЛЕ удаления %d звеньев\n", SP.num()); getchar();

    SP.print("DOWN");    //,1);
    getchar();
};
ddd:

// удалить список из памяти

```

```
// delete SP;
printf("ycëëëë\n");

return 0;
};
```

Результат работы теста

```
прочитали 1 <111111111111111>
прочитали 2 <222222222222222>
прочитали 3 <333333333333333>
прочитали 4 <444444444444444>
прочитали 5 <555555555555555>
```

```
first=<00842E00> tek=<00842F80> ends=<00842F80>
```

```
ADD=<00842E00> ZVENO=<00000000><00842E60><111111111111111>
```

```
ADD=<00842E60> ZVENO=<00842E00><00842EC0><222222222222222>
```

```
ADD=<00842EC0> ZVENO=<00842E60><00842F20><333333333333333>
```

```
ADD=<00842F20> ZVENO=<00842EC0><00842F80><444444444444444>
```

```
ADD=<00842F80> ZVENO=<00842F20><00000000><555555555555555>
```

```
***** EOS (ends) ***** нажмите Enter
```

```
first=<00842E00> tek=<00842F80> ends=<00842F80>
```

```
ADD=<00842F80> ZVENO=<00842F20><00000000><555555555555555>
```

```
ADD=<00842F20> ZVENO=<00842EC0><00842F80><444444444444444>
```

```
ADD=<00842EC0> ZVENO=<00842E60><00842F20><333333333333333>
```

Кафедра программирования и информационных технологий

```
ADD=<00842E60> ZVENO=<00842E00><00842EC0><222222222222222>
ADD=<00842E00> ZVENO=<00000000><00842E60><111111111111111>
***** EOS (first) *****
    нажмите Enter
распечатка списка в прямом направлении .....
num=0001 zap=<111111111111111>
num=0002 zap=<222222222222222>
num=0003 zap=<333333333333333>
num=0004 zap=<444444444444444>
num=0005 zap=<555555555555555>
***** EOS (ends) *****
    нажмите Enter
```