Класичний приватний університет

Кафедра програмування та інформаційних технологій

Борю С.Ю

МЕТОДИЧНІ ВКАЗІВКИ
ДО ВИКОНАННЯ КУРСОВОЇ РОБОТИ З ДИСЦИПЛІНИ

Об‘єктно-орієнтоване програмування

для студентів денного і заочного відділення спеціальності
7.080404 Програмне забезпечення автоматизованих систем
7.080200 - Системний аналіз

Запоріжжя, 2010

Методичні вказівки по виконанню курсової роботи по курсу «об‘ектно - орієнтоване програмування» для студентів денного і заочного відділення спеціальності "Програмне забезпечення автоматизованих систем" і „Системний аналіз і управлення” /сост. Борю С.Ю. - Запоріжжя: Класичний приватний університет, 2010 р. - с. 43

Дані методичні вказівки включають приклади розрахунків, варіанти курсової роботи і рекомендовані джерела, які можна використовувати при виконанні роботи.

Метою курсової роботи є закріплення і поглиблення відомий, одержаних на лекціях, практичних заняттях і в процесі самостійного вивчення матеріалу по курсу «об‘ектно - орієнтоване програмування».

Курсова робота призначена для перевірки теоретичних і практичних відомий студентів; для відробітку навиків самостійного вивчення предмету, роботи з літературою і програмним забезпеченням, уміння логічно і послідовно висловлювати засвоєний матеріал.

Содержание

Загальні вимоги до виконання курсової роботи.

Для виконання роботи необхідно:
вивчити теоретичний матеріал;

- ознайомитися з приведеним прикладом;
- вибрати варіант по наступній формулі - номер варіанту = залишок від ділення номера Вашій заченой книжці на число 24.
- виконати роботу;
- оформити звіт по роботі, використовуючи текстовий процесор MS Word або подібний ;
- захист контрольної роботи припускає наявність електронного варіанту звіту і працюючої програми.

Вимоги до оформлення звіту по роботі.

1. Зміст роботи оформити як автоматичний зміст з вказівкою сторінок і заповнювача.
2. Передбачити звичайні виноски в тексті на використані джерела (література, програмне забезпечення).
3. Список джерел оформити як нумерований список.
4. У роботі передбачити наступні розділи:
5. Титульний лист
6. Зміст
7. Теоретичні відомості
8. Порядок виконання роботи
9. Список використаних джерел.
10. Створити наступні колонтитули для розділів:
11. верхній - зліва Ваша ФІО, справа - назва розділу;
12. нижній - зліва номер варіанту, справа - нумерація сторінок;
13. титульний лист - колонтитули відсутні.
14. Теоретичні відомості оформляються в текстовому процесорі Word, з описом типів і форматів даних, видів посилань, формул, синтаксису функцій, побудови діаграм.
15. Описати порядок виконання роботи.

Розробка простих класів

Варіанти завдань

1. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:

Об’єкт «комплексні числа». Операції визначаються по загальноприйнятим формулам. Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод add і sub - складання і віднімання двох комплексних чисел. Реалізуйте перевантаження операторів привласнення, складання і вичитатія.

2. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:

Об’єкт «комплексні числа». Операції визначаються по загальноприйнятим формулам. Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод mul і div - множення і ділення двох комплексних чисел. Реалізуйте перевантаження операторів привласнення, умноження і ділення.

3. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:

Об’єкт «комплексні числа». Операції визначаються по загальноприйнятим формулам. Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод toString і EQ – перетворення числа у символьний рядок і порівняння двох комплексних чисел. Реалізуйте перевантаження операторів привласнення і конструктор копіювання.

4. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:

   Об’єкт «комплексні числа». Операції визначаються по загальноприйнятим формулам. Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод MOD і ConJ – знаходження

модуля числа і отримання пов’язаного числа. Реалізуйте перевантаження операторів привласнення і конструктор копіювання.

5. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
   Об’єкт «вектор на площині» заданий в системі декартових координат. Початок вектора розташовано на початку координат. Операції визначаються згідно загальноприйнятих формул лінійної (векторної) алгебри. Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод add і sub - складання і віднімання двох векторів. Реалізуйте перевантаження операторів привласнення і конструктор копіювання.

6. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
   Об’єкт «вектор на площині» заданий в системі декартових координат. Початок вектора розташовано на початку координат. Операції визначаються згідно загальноприйнятих формул лінійної (векторної) алгебри. Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод mul і Ang – скалярного множення двох векторів і знаходження кута між двома векторами. Реалізуйте перевантаження операторів привласнення та умноження

7. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
   Об’єкт «вектор на площині» заданий в системі декартових координат. Початок вектора розташовано на початку координат. Операції визначаються згідно загальноприйнятих формул лінійної (векторної) алгебри. Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод print і EQ – перетворення координат вектора у символьний рядок і порівняння двох векторів. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

8. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
   Об’єкт «прямокутний трикутник, заданий довжинами катетів». Конструктор повинен дозволити створювати об’єкти без та з початковою

ініціалізацією. Реалізувати методи знаходження гіпотенузи і площі трикутника. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

9. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
   Об’єкт «прямокутний трикутник, заданий довжинами катета і гіпотенузи». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати методи знаходження периметра і площі трикутника. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

10. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «раціональний не скоротний дріб, представлений парою цілих чисел». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод add і sub - складання і віднімання двох дробів. Реалізуйте перевантаження операторів привласнення, складання і вичитатія.

11. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «раціональний не скоротний дріб, представлений парою цілих чисел». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод mul і div - множення і ділення двох дробів. Реалізуйте перевантаження операторів привласнення, умноження і ділення.

12. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «раціональний не скоротний дріб, представлений парою цілих чисел». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод toString і EQ – перетворення дробі у символьний рядок і порівняння на двох дріб. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

13. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «множина цілих чисел заданої потужності». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією на основі заданого масиву. Потужність множини задається при створенні об’єкту. Реалізувати метод, що відповідає на питання, чи містить множину вказане число, і метод друку на консоль елементів множини. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

14. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «стек цілих чисел». Конструктор задає розмір стека. Реалізувати метод PUSH і POP - помістити число в стек і витягнути із стека. Методи повинні повертати логічні значення true або false залежно від успіху або невдачі виконаної операції з стеком. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

15. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «таблиця що містить два стовпці - номер телефону і прізвище абонента». Конструктор задає розмір таблиці. Реалізувати метод ADD та GET - помістити номер телефону та прізвище абонента у таблицю і повідомити номер телефону (або його відсутність) для заданого імені абонента. Методи повинні повертати логічні значення true або false залежно від успіху або невдачі виконаної операції. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

16. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «раціональний не скоротний дріб, представлений парою цілих чисел». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод mul і div - множення і ділення двох дробів. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

17. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «раціональний не скоротний дріб, представлений парою цілих чисел». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод toString і EQ – перетворення дробі у символьний рядок і порівняння на двох дріб. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

18. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «вектор на площині» заданий в системі декартових координат. Початок вектора розташовано на початку координат. Операції визначаються згідно загальноприйнятих формул лінійної (векторної) алгебри. Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод mul і Ang – скалярного множення двох векторів і знаходження кута між двома векторами. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

19. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «вектор на площині» заданий в системі декартових координат. Початок вектора розташовано на початку координат. Операції визначаються згідно загальноприйнятих формул лінійної (векторної) алгебри. Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод print і EQ – перетворення координат вектора у символьний рядок і порівняння двох векторів. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

20. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «комплексні числа». Операції визначаються по загальноприйнятим формулам. Конструктор повинен дозволити створювати об’єкти без та з

початковою ініціалізацією. Реалізувати метод mul і div - множення і ділення двох комплексних чисел. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

21. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
Об’єкт «комплексні числа». Операції визначаються по загальноприйнятим формулам. Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод toString і EQ – перетворення числа у символьний рядок і порівняння двох комплексних чисел Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

22. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
Об’єкт «прямокутний трикутник, заданий довжинами катетів». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати методи знаходження гіпотенузи і площі трикутника. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

23. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
Об’єкт «прямокутний трикутник, заданий довжинами катета і гіпотенузи». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати методи знаходження периметра і площі трикутника. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

24. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
Об’єкт «раціональний не скоротний дріб, представлений парою цілих чисел». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод add і sub - складання і

віднімання двох дробів. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

25. Розробити клас, набір методів (конструктор, деструктор і вказані методи) для програмної моделі заданого об’єкту. Опис об’єкту і його основних властивостей приводиться нижче. Привести фрагмент програми (int main), що використовує об’єкти розробленого класу:
    Об’єкт «раціональний не скоротний дріб, представлений парою цілих чисел». Конструктор повинен дозволити створювати об’єкти без та з початковою ініціалізацією. Реалізувати метод mul і div - множення і ділення двох дробів. Реалізуйте перевантаження операторів привласнення та ввиденя-виведення.

Зміст звіту

1. Постановка завдання.
2. Опис класу, методи для роботи з об’єктом.
3. Функція main() - що тестує всі розроблені методи.
4. Результати виконання тестуючої програми.

Приклади подібних *учбових* реалізацій можна подивитися в додатку 2 і 3

Список рекомендованих джерел

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.
4. Программирование на С++: Учебное пособие / В. П. Аверкин и др.; Под ред. проф. А. Д. Хомоненко.- СПб.:КОРОНА принт, 1999. – 256 с.
5. Страуструп Б. Язык программирования С++, 3-е изд./ Пер. с англ.-СПб.; М.:"Невский Диалект" – "Издательство БИНОМ",1999 г. – 991 с.
6. Шилдт Г. Самоучитель С++, 3-е изд./ Пер. с англ.-СПб.: BHV-СПб., 1998 г. – 800 с.
7. Шилдт Г. Программирование на Borland C++ для профессионалов / Пер. с англ.- Мн.: ООО «Попурри», 1998 г. - 800 с.

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.

КУРСОВА РОБОТА № _____

з дисципліни _____
на тему
_____
_____

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

Перевірив:
_____
(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, населенний 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 СЕДУЮЩЕГО 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[1]; 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 - конактенитрованная 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 - конактенитрованная 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
        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_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 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
        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;      // передыдущее 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 изменяеется)
{ 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,argc[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 ааааааа 111***********");
   SP.add_ends("Добавка в конец ааааааа 222***********");
   SP.add_ends("Добавка в конец ааааааа 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 ааааааа 111***********");
   SP.add_first("Добавка в начало ааааааа 222***********");
   SP.add_first("Добавка в начало ааааааа 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("усёёёё\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>
```

```
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
```