# САОД. Задание 4

## Темы

* Классы с динамическими полями;
* Операторы преобразования типа и другие операции;
* Конструктор копирования;
* Операции с памятью;

## Задания

1. Подготовительные действия:
   1. Откройте презентацию второй лекции. (*Groups\\_ИСАУ\3 курс\САОД*).
   2. Если вы не слушали лекцию 2, то познакомьтесь с ней, начиная с раздела “*Виды конструкторов*”, в котором разбирается класс ***Str***.
   3. Создайте новый проект Strings и добавьте в него операции ввода-вывода на консоль. ***Не забывайте делать это на рабочем столе.***
   4. Добавьте в проект класс Str. Это можно, как и раньше, сделать при помощи мастера добавления класса, а можно просто добавив заголовочный файл “Str.h”.  
        
      Если воспользовались мастером добавления класса, то удалите файл Str.cpp. Наш класс будет состоять ТОЛЬКО из inline методов.
   5. Скопируйте описание класса Str из лекции в заголовочный файл. Включите Str.h в модуль Strings.cpp, где расположена main.   
        
      Объявите объект класса Str в main и убедитесь, что проект собирается и выполняется. Если необходимо, устраните ошибки.
2. Добавьте в класс конструктор инициализации z-строкой (из лекции).   
     
   Проинициализируйте объявленный в main объект Str каким-нибудь строковым литералом. Например, Str s = “123”.   
     
   Убедитесь, что проект компилируется и собирается. При необходимости устраните ошибки.
3. Попробуйте вывести объявленный объект на консоль. Скорее всего вы увидите сообщение, что оператор << для выходного потока и Str не определен!  
     
   Добавьте в класс оператор преобразования типа   
   operator const char \*()const { return m\_pszText; }

*Он “учит” компилятор преобразовывать объекты Str в const char \*.*

Убедитесь, что теперь вывод объектов Str на консоль (*как и в другие потоки вывода*) возможен.

Убедитесь также, что к объектам Str можно применять функцию strlen и другие функции, которые получают параметрами const char \*.   
*Например, выведите strlen(s) на консоль или сохраните в целочисленной переменной.*

1. Добавьте перед main функцию void Test(Str s) {}. Обратите внимание, функция ничего не делает, только получает параметр и завершается.  
     
   После строк вывода на консоль добавьте вызов функции, а параметром передайте ваш объект.  
     
   Продублируйте строки вывода после вызова функции. Выполните программу.   
     
   Скорее всего, при выполнении появятся ошибки. Почему?
2. Добавьте в определение класса конструктор копирования из лекции. Убедитесь, что теперь программа работает без ошибок! Объясните для себя причину.
3. Перед **return 0**; в main добавьте фрагмент кода:  
   {

Str t;

t = s;

}  
  
После блока продублируйте строки вывода. Выполните программу.   
Скорее всего, при выполнении появятся ошибки. Почему?

1. Добавьте в определение класса оператор присваивания из лекции. Убедитесь, что теперь программа работает без ошибок! Объясните для себя причину.

Немного поиграйте с присваиванием. Убедитесь, что компилятор разрешает   
 t = s = t;

и подобные инструкции. Что при этом происходит?

1. Самостоятельно разработайте оператор +=, который добавляет в конец строки объекта, строку, переданную в качестве параметра.  
    Str & operator +=(const char \*sz)  
   ***Важно!*** *Сначала продумайте как управлять памятью при выполнении такой операции:*
   1. *Почему объект будет иметь достаточно места для хранения новой строки?*
   2. *Какова судьба блока памяти, которой владеет объект на момент начала операции?*
   3. *Какова судьба блока памяти, который передан параметром?*

*Примените приемы из уже имеющихся методов. Как узнать длину строки, как выделить память, как копировать строки…*

Проверьте работоспособность вашей реализации в main. Например, подготовьте 2 объекта Str с разными начальными значениями. Выведите их на консоль. Добавьте один объект к другому (*почему это возможно? Ведь мы учили добавлять только z-строки, а добавили объект.*) и повторите вывод.

Добейтесь чтобы программа работала без ошибок.

1. Самостоятельно разработайте оператор +, который получает новый объект, равный сумме двух строк.   
    Str operator +(const char \*sz) const;  
     
   *Опять, выгодно добавлять именно строку, а не объект. Почему*?  
   *Обратите внимание, теперь оператор возвращает не ссылку на объект, а именно объект. Ведь возникает новый экземпляр класса (сумма строк). Исходный объект не изменяется* (const)!

Максимально используйте уже имеющиеся методы класса. Ведь обращение к ним “ничего не стоит”, они все ***inline***.

Проверьте работоспособность нового оператора.

**Теперь мы научились**:

* Создавать классы с динамическими полями данных;
* Создавать для них деструктор и конструктор копирования;
* Разрабатывать операторы преобразования типа (*у них не указывается тип возвращаемого значения*);
* Управлять динамической памятью, выделять ее и освобождать, так чтобы не возникало “утечек памяти”.

# Отчет

Подготовьте в Word краткий (*не более 1 стр.*) отчет о выполнении задания.

В отчете: опишите особенности операторов преобразования типа (чем отличаются от объявления других операторов), опишите применения квалификатора const в C++, опишите логику оператора += для класса Str.

Отчет и архив решения поместите как ответ на задание.