Практическая работа №4. Строки

Унаследуйте свой класс от базового класса «Строка». Реализуйте работу функций: конструкторов, конструктора копий, деструктора, operator=, как в базовом классе, так и в наследнике. Добавьте в наследник работу функции, указанной в варианте.

Вариант	Функция
1.	Написать функцию поиска LastIndexOf() последнего вхождения произвольной подстроки в строке.
2.	Написать функцию поиска IndexOf() первого вхождения произвольной подстроки в строке.
3.	Написать функцию, вводящую аналог операции “<” для сравнения двух строк (сравнение в алфавитном порядке).
4.	Написать функцию string_to_number(), проверяющую, является ли содержимое строки числом, и возвращающую это число в случае утвердительного ответа.
5.	Написать функцию, определяющую количество вхождений некоторой подстроки
6.	Написать функцию compare(), проверяющую совпадают ли две переданные ей строки
7.	Получить символ, который повторяется наибольшее количества раз. Если таких несколько, то вернуть массив из этих символов
8.	Написать функции, реализующие код Цезаря (и кодирование, и декодирование): буква заменяется на следующую за ней в алфавите (“test”->”uftu”)
9.	Написать функцию, проверяющую, является ли строка палиндромом
10.	Написать функцию, подсчитывающую число гласных
11.	Написать функцию endswith(), проверяющую, заканчивается ли строка переданным массивом символов
12.	Получить все цифры в строке в виде массива целых чисел
13.	Подсчитать количество знаков препинания (.,:;-) в исходной строке
14.	Написать функцию toupper(), преобразующую строку к верхнему регистру
15.	Подсчитать количество слов в строке. Словом будем считать последовательность, не содержащую цифр, имеющую длину не менее 3-х символов. Слова отделены друг от друга знаками препинания и пробелами.
16.	Подсчитать количество пар символов, в которых за согласной следует гласная.
17.	Написать функцию, подсчитывающую количество заглавных букв.
18.	Строку мысленно разбить на последовательности по 5 символов. Подсчитать количество пятерок символов, для которых среднее количество гласных больше среднего для всей строки.
19.	Написать функцию trim(), удаляющую все пробелы в начале и конце строки
20.	Написать функцию reverse(), заменяющую порядок символов в строке на

	обратный
21.	Написать функцию, удаляющую лишние пробелы в тексте
22.	Написать функцию insert(), вставляющую подстроку в строку в место, обозначенное индексом index
23.	Написать функцию startswith(), проверяющую, начинается ли строка переданным массивом символов
24.	Определить число символов в строке, которые больше заданного символа ("больше" в смысле алфавитного порядка). Учесть, что регистр не важен.
25.	Подсчитать количество слов, которые начинаются и заканчиваются гласной.
26.	Определить, является ли строка допустимым идентификатором в C
27.	Получить индекс самого длинного слова в строке.
28.	В строке присутствуют скобки (). Проверить, что они корректно расставлены - для каждой открывающей есть закрывающая, и они расположены в правильном порядке. Пример неверной записи: ")()(", "(())())"
29.	Для всех слов в строке посчитать сумму значений кодов ASCII для символов, входящих в слово. Вернуть наибольшее значение.
30.	Проверить, идет ли в строке пробел после каждого знака препинания.
31.	Написать функцию tolower(), преобразующую строку к нижнему регистру
32.	Получить индекс последовательности с максимальным числом одинаковых символов. Если их несколько, вернуть индекс начала самой первой
33.	Подсчитать количество слов, в которых равное число гласных и согласных
34.	Написать функцию, которая принимает строку и возвращает новую, в которой все гласные переставлены в начало массива
35.	Получить индекс максимального по длине участка идущих подряд согласных

Пример кода:

```
#include <iostream>

using namespace std;

class BaseString
{
protected:
	char* p;
	int len;
	int capacity;
public:
	BaseString(char* ptr)
	{
```

```
	cout<<"\nBase Constructor 1\n";
	len = strlen(ptr) + 1;
	capacity = 256;
	p = new char[capacity];
	for(int i=0;i<len;i++)
	{
		p[i] = ptr[i];
	}
	p[len] = '\0';
}

BaseString(int Capacity = 256)
{
	cout<<"\nBase Constructor 0\n";
	capacity = Capacity;
	p = new char[capacity];
	len = 0;
}

~BaseString()
{
	cout<<"\nBase Destructor\n";
	if(p!=NULL)
		delete[] p;
	len = 0;
}

int Length() {return len;}
int Capacity() { return capacity; }
//char* get() {return p;}
char& operator[](int i) {return p[i];}


BaseString& operator=(BaseString& s)
{
	cout<<"\nBase Operator = \n";
	//if(p!=NULL)
		//delete[] p;
	len = s.Length();
	p = new char[s.capacity];
	capacity = s.capacity;
	for(int i=0;i<s.Length();i++)
	{
		p[i] = s[i];
	}
	//strcpy(p, s.get());
	p[len-1] = '\0';
	return *this;
}

BaseString(BaseString& s)
{
	cout<<"\nBase Copy Constructor\n";
```

```
            //if(p!=NULL)
                  //delete[] p;
            len = s.Length();
            p = new char[s.capacity];
            capacity = s.capacity;
            for(int i=0;i<s.Length() - 1;i++)
            {
                  p[i] = s[i];
            }
            p[len-1] = '\0';
      }

      virtual void print()
      {
            int i=0;
            while(p[i]!='\0')
            {
                  cout<<p[i];
                  i++;
            }
      }
};

class String: public BaseString
{
public:
      String(char* ptr) : BaseString(ptr) { cout << "\nDerived
Constructor 1\n"; }

      String(int Capacity = 256): BaseString(Capacity) { cout <<
"\nDerived Constructor 0\n"; }


      ~String()
      {
            cout << "\nDerived Destructor\n";
      }

      String& operator=(String& s)
      {
            cout << "\nDerived operator = \n";
            BaseString::operator=((BaseString&)s);
            return *this;
      }

      String(String& s)
      {
            cout << "\nDerived Copy Constructor\n";
            //if(p!=NULL)
                  //delete[] p;
            len = s.Length();
            p = new char[s.capacity];
            capacity = s.capacity;
```

```
            for (int i = 0; i < s.Length() - 1; i++)
            {
                  p[i] = s[i];
            }
            p[len - 1] = '\0';
      }

      String& operator+(String& s)
      {
            int len1 = len + s.Length() - 1;

            int l1 = len;
            int i = l1-1;
            while (s[i - l1] != '\0')
            {
                  p[i] = s[i - l1+1];
                  i++;
            }
            p[i] = '\0';
            len = len1;
            return *this;
      }
};

int _tmain(int argc, _TCHAR* argv[])
{
      if (true)
      {
            String s("test");
            s.print();
            String s1 = s;
            s1.print();
            String s2;
            s2 = s;
            s2 = s + s1;
            s2.print();
            s1 = s2 + s;
            s1.print();
      }
      char c; cin>>c;
      return 0;
}
```