Первое задание по практикуму для самостоятельного программирования дома

Напишите программу, в которой определяются классы «строка», «блочная строка», «выборка», «блочная выборка», «поток вывода» и другие вспомогательные классы по желанию. Блочной строкой называется массив строк. Над блочными строками должны быть допустимы следующие операции:

- Должно быть задано правило преобразования строки в блочную строку — массив, состоящий из одного элемента: `"abc"` → `{"abc"}`.
- «`+`»: при сложении двух блочных строк соответствующие массивы должны конкатенироваться. Например, `{"abc"} + {"c", "b", "a"}` → `{"abc", "c", "b", "a"}`.
- «`*`»: при умножении блочной строки на целое число должна получиться блочная строка, состоящая из копий исходной: `{"a", "b"} * 3` → `{"a", "b", "a", "b", "a", "b"}`. Аналогично с `3 * {"a", "b"}`.
- «`[]`»: если `b` — блочная строка, то `b[n]` должно быть вычислено как *строка* в массиве блочной строки, имеющая индекс `n`.
- «`()`»: если `b` — блочная строка, то `b(n)` должно быть вычислено как *символ* блочной строки, имеющий индекс `n`. Нумерация символов блочной строки сквозная: например, символ блочной строки `{"abc", "def"}` с индексом 0 равен `a`, символ с индексом 1 равен `b`, символ с индексом 3 равен `d`, а символ с индексом 5 равен `f`.

Из приведённых правил следует, что сумма двух строк является блочной строкой — массивом из двух элементов, а произведение строки на целое число `n` — блочной строкой, представляющей собой массив из `n` одинаковых строк.

Выборкой называется функция, описываемая массивом из целых чисел. В результате применения этой функции к строке должна получиться другая строка, состоящая из символов исходной строки, взятых в порядке, задаваемым массивом – выборкой. Например, если выборка задаётся массивом `{2, 0, 1}` и применяется к строке `"abc"`, то должна получиться строка `"cab"`. Если индекс выходит за границы строки, то в соответствующей позиции должен быть поставлен символ `?`: применение выборки `{-1, 2, 3, 1}` к строке `"abc"` даёт строку `"?c?b"`. Если выборка применяется к блочной строке, то используется сквозная нумерация её символов: например, применение выборки `{6, 5, 4, 3, 2, 1, 0}` к блочной строке `{"abc", "def"}` даст строку `"?fedcba"`.

Блочной выборкой называется массив из выборок. Аналогично строкам должно быть задано правило преобразования выборки в блочную выборку, операции `+` и `*`, конкатенирующие массивы блочных выборок, а также операция индексирования `[]`, дающая выборку из массива с заданным индексом. Применение блочной выборки к блочной строке обозначается операцией `()` и приводит к блочной строке, каждая составная строка которой является результатом применения соответствующей выборки из блочной выборки к исходной блочной строке. Например, в результате применения блочной выборки `{{4, 3, 2, 1, 0}, {3, 5, 1}, {0}}` к блочной строке `{"abc", "def"}` должна получиться блочная строка `{"edcba", "dfb", "a"}`.

Должно быть предусмотрено задание блочных выборок, являющихся результатом последовательного применения ранее определённых блочных выборок. Например, если `f` и `g` — блочные выборки, то выражение `f(g)` является блочной выборкой, применение которой к блочной строке `s` должно быть эквивалентно вычислению выражения `f(g(s))`. Вычисления не должны быть ленивыми, суперпозиция `f(g)` должна вычисляться сразу же. Например, для `f = {{4, 3, 2, 1, 0}, {3, 5, 10}, {0}}`, `g = {{1, 2, 1}, {4, 5, 5}}` суперпозиция является блочной выборкой `f(g) = {{5, 4, 1, 2, 1}, {4, 5, -1}, {1}}`.

Все классы должны быть описаны в отдельном пространстве имён. В этом же пространстве имён должна находиться глобальная переменная – поток вывода блочных строк и выборок с двумя манипуляторами, один из которых выводит блочную строку в виде массива — например, в приведённом выше виде — а другой выводит её как длинную строку — конкатенацию составляющих её строк. Вывод должен производиться на стандартный поток вывода. Также в этом пространстве имён должен быть описан класс «поток вывода», имеющий конструктор с одним аргументом типа `std::ostream` и позволяющий выводить блочные строки и выборки в заданный поток, возможно, с указанием описанных выше манипуляторов.

В функцию `main` поместите свой код, показывающий работоспособность программы. Пример:

```
std::string s = "abc";
strings::String a, b = "a", c = s;
strings::BlockString bstr = a + b * 1 + 2 * c;
// Prints {"", "a", "abc", "abc"}:
```

```
strings::cout << strings::array << bstr << '\n';
strings::cout << strings::plain << bstr << '\n'; // Prints aabcabc
strings::cout << bstr(3) << ' ' << bstr[3] << '\n'; // c abc

std::ofstream f("a.txt");
string::OStream fout(f);
fout << strings::plain << bstr << '\n'; // Prints aabcabc to file a.txt
fout.close(); // Closes file a.txt

std::vector<std::vector<int>> v = {{4, 3, 2, 1, 0}, {3, 5, 10}, {0}};
strings::BlockSample f(v);
strings::cout << strings::array << f(bstr) << '\n'; // {"acbaa", "cb?", "a"}
strings::cout << strings::plain << f(bstr) << '\n'; // acbaacb?a

v = {{1, 2, 1}, {4, 5, 5}};
strings::BlockSample g(v);
strings::cout << strings::array << g(bstr) << '\n'; // {"aba", "abb"}
strings::cout << strings::plain << g(bstr) << '\n'; // abaabb

strings::BlockSample fg = f(g); // fg = {{5, 4, 1, 2, 1}, {4, 5, -1}, {1}}
// Prints some representation of block samples:
strings::cout << f << '\n' << g << '\n' << fg << '\n';
// Prints {"baaba", "ab?", "a"}:
strings::cout << strings::array << fg(bstr) << '\n';
strings::cout << strings::plain << fg(bstr) << '\n'; // baabaab?a

strings::Sample shift_left(std::vector<int>({1, 2, 3, 4, 5, 6, 7, 8, 0}));
strings::BlockSample h = shift_left(fg);
// h = {{4, 1, 2, 1, 4, 5, -1, 1, 5}}
strings::cout << strings::array << h(c) << '\n'; // {"?bcb???b?"}
strings::cout << strings::plain << h(c) << '\n'; // ?bcb???b?

shift_left = std::vector<int>({1, 2, 0});
strings::Sample shift_right(std::vector<int>({2, 0, 1}));
f = shift_left * 1 + 2 * shift_right;
// f = {{1, 2, 0}, {2, 0, 1}, {2, 0, 1}}
strings::cout << strings::array << f(s) << '\n'; // {"bca", "cab", "cab"}
strings::cout << strings::plain << f(s) << '\n'; // bcacabcab

a = "b";
bstr = (shift_right + shift_left(shift_right))(a + b * 2);
strings::cout << strings::array << bstr << '\n'; // {"aba", "baa"}
strings::cout << strings::plain << bstr << '\n'; // ababaa
```

Требования:

1. Если для некоторого класса в программе автоматически генерируемые конструктор копирования, операция присваивания или деструктор работают некорректно, они должны быть заменены на пользовательские.

2. При возникновении ошибки нужно сообщить об этом пользователю. Форма сообщения может быть произвольной, например, через компилятор или с помощью вывода на экран.

3. Программа должна быть разбита на несколько модулей. Один из них должен представлять .cpp-файл, содержащий функцию `main`, другие модули должны быть представлены в виде пар .hpp- и .cpp-файлов (части имён до «.» совпадают), первый из которых содержит объявления класов и функций, второй — их описания. Заголовочные .hpp-файлы должны быть защищены от повторного включения (`#ifndef` - `#define`; ~~`#pragma`~~).

Срок сдачи: 23:59, 15.03.2024