Перегрузка бинарных операторов для класса String

Класс mystring

- Наследуется от std::string для доступа к перегруженному конструктору и функциям append(“”), size() и compare(“”);

```
class mystring : public std::string {
 public:
  mystring();
  mystring(const char* str);
  mystring(const mystring& str);
  mystring& operator+=(const mystring& concat);
  mystring& operator+=(const char* concat);
};
```

Конструкторы mystring

- Предусмотрена инициализация mystring пустой строки, от строкового литерала и от другого mystring

```
mystring::mystring() : std::string() {
}

mystring::mystring(const char* str) : std::string(str) {
}

mystring::mystring(const mystring& str) : std::string(str) {
}
```

Конструкторы mystring

- Примеры объявлений

```
mystring s; // mystring()
mystring ss("try catch!"); // mystring(const char* str)
mystring sss = "or try escape!"; // mystring(const char* str)
const auto ssss = sss; // mystring(const mystring& str)
```

Перегрузка операторов mystring

Оператор присвоенного сложения:

```
mystring& mystring::operator+=(const mystring& concat) {
  this->append(concat);
  return *this;
}
mystring& mystring::operator+=(const char* concat) {
  this->append(concat);
  return *this;
}
```

Выполняет конкатенацию concat-строки в this-строку с помощью append()

Перегрузка операторов mystring

Оператор присвоенного сложения:

```
mystring s = "Sherlock", s1 = " Holmes\n";
s += s1;
std::cout << s;
```

Вывод: “Sherlock Holmes”

Перегрузка операторов mystring

Оператор сложения:

```
mystring operator+(const mystring& receive, const mystring& send) {
  mystring newbie;
  newbie.append(receive).append(send);
  return newbie;
}
mystring operator+(const mystring& receive, const char* send) {
  mystring newbie;
  newbie.append(receive).append(send);
  return newbie;
}
mystring operator+(const char* receive, const mystring& send) {
  mystring newbie;
  newbie.append(receive).append(send);
  return newbie;
}
```

Выполняет сложение строк receive и send. Метод append() выполняется дважды для нового mystring.

Перегрузка операторов mystring

Оператор сложения:

- Можно использовать operator+ как метод mystring в двух случаях. А случай (`const char* receive, const mystring& send`) оставить как free-функцию.

- В том примере ввиду отличия только одних параметров для наглядности оставим этот оператор полностью свободным

Перегрузка операторов mystring

Оператор сложения:

```
// mystring operator+(const char *receive, const mystring &send)
mystring ms1 = " chloride ";
auto res1 = " ferrum " + ms1 + " magnezite ";
std::cout << res1 << '\n';
// mystring operator+(const char *receive, const mystring &send)
mystring ms2 = " ferrum ";
auto res2 = ms2 + " chloride " + " magnezite ";
std::cout << res2 << '\n';
// mystring operator+(const mystring &receive, const mystring &send)
mystring ms3 = " ferrum ", ms3_(" magnezite ");
auto res3 = ms3 + " chloride " + ms3_;
std::cout << res3 << '\n’;
```

Вывод: ” ferrum chloride magnezite “

Перегрузка операторов mystring

Оператор проверки равенства:

```
bool operator==(const mystring& first, const mystring& second) {
  if (first.size() != second.size())
    return false;
  return first.compare(second) == 0;
}
bool operator==(const mystring& first, const char* second) {
  return first.compare(second) == 0;
}
bool operator==(const char* first, const mystring& second) {
  return second.compare(first) == 0;
}
```

Этот оператор использует std::string-метод сравнения строк compare(). В первом случае мы можем сразу проверить равенство через длины строк size(), не прибегая к compare()

Перегрузка операторов mystring

Оператор проверки неравенства:

```
bool operator!=(const mystring& first, const mystring& second) {
  if (first.size() == second.size())
    return false;
  return first.compare(second) != 0;
}
bool operator!=(const mystring& first, const char* second) {
  return first.compare(second) != 0;
}
bool operator!=(const char* first, const mystring& second) {
  return second.compare(first) != 0;
}
```

Аналогично с operator== мы можем использовать compare и для проверки неравенства.

Перегрузка операторов mystring

Примеры использования:

```
// bool operator==(const mystring &first, const mystring &second)
mystring ms4 = "false";
if (ms4 == mystring("false")) {
  std::cout << "ms4 == mystring(\"false\")\n";
}
// bool operator==(const mystring &first, const char* second)
mystring ms5 = "false";
if (ms5 == "false") {
  std::cout << "ms5 == \"false\"\n";
}
// bool operator==(const char *first, const mystring& second)
mystring ms6 = "false";
if ("false" == ms6) {
  std::cout << "\"false\" == ms6\n";
}
```

```
// bool operator!=(const mystring &first, const mystring &second)
mystring ms7 = "false";
if (ms7 != mystring("true")) {
  std::cout << "ms7 != mystring(\"true\")\n";
}
// bool operator!=(const mystring &first, const char* second)
mystring ms8 = "false";
if (ms8 != "true") {
  std::cout << "ms8 != \"true\"\n";
}
// bool operator!=(const char *first, const mystring& second)
mystring ms9 = "false";
if ("true" != ms9) {
  std::cout << "\"true\" != ms9\n";
}
```