ФИО: Ребдев Павел Александрович
Группа: 5130904/30008
Лабораторная работа: «Класс для работы со строками»

Постановка задачи

Создать класс String, содержащий в себе строку, её характеристики и методы для работы с ней

Исходные тексты программы

Файлы с исходными текстами лабораторной работы (полагаем <ROOT> для папки в котором располагаются исходные тексты):

./<ROOT>/main.cpp

```
#include <iostream>
#include <cstddef>
#include "String.h"

int main()
{
  rebdev::String str1;
  const char * str2Char = "string2";
  rebdev::String str2(str2Char);
  rebdev::String str3(str2);
  const char * str4Char = "string4";
  rebdev::String str4(str4Char);

  std::cout << "compare 2 and 3: " << str2.compare(str3) << '\n';
  std::cout << "compare 2 and 4: " << str2.compare(str4) << '\n';

  rebdev::String str7(str2.insert(2, "ok"));
  rebdev::String str8(str7 + str3);
  size_t i = 0;

  std::cout << "insert ok in 2:";
  while (str7[i] != '\0')
  {
    std::cout << str7[i];
    i += 1;
  }
  i = 0;

  std::cout << "7 + 3:";
  while (str8[i] != '\0')
  {
    std::cout << str8[i];
    i += 1;
  }
  i = 0;

  return 0;
}
```

./<ROOT>/String.h

```
#ifndef STRING_H
#define STRING_H
#include <cstddef>
namespace rebdev
```

```
{
  class String
  {
    public:
      String();
      String(const char * str);
      String(const String& str);
      String(String&& str) noexcept;
      ~String ();

      String& operator= (const char* str);
      String& operator= (String&& str) noexcept;
      char& operator[] (size_t pos) const;
      String& append (const String& str);
      String& operator+ (const String& str);
      String& insert (size_t pos, const char* str);
      int compare (const String& str);
    private:
      size_t size_;
      size_t capacity_;
      char * pointer_;
  };
}
#endif
```

./<ROOT>/String.cpp

```
#include "String.h"

rebdev::String::String():
  size_(0),
  capacity_(0),
  pointer_(nullptr)
{};

rebdev::String::String(const char * str):
  size_(0),
  capacity_(0),
  pointer_(nullptr)
{
  if (pointer_ != nullptr)
  {
    delete[] pointer_;
  }

  size_ = 0;
  while (str[size_] != '\0')
  {
    size_ += 1;
  }

  pointer_ = new char[size_];
  for (size_t i = 0; i < size_; ++i)
  {
    pointer_[i] = str[i];
```

```
  }
  capacity_ = (size_ * sizeof(char));
};

rebdev::String::String(const String& str):
  size_(0),
  capacity_(0),
  pointer_(nullptr)
{
  while (str[size_] != '2')
  {
    size_ += 1;
  }

  pointer_ = new char[size_];
  for (size_t i = 0; i < size_; ++i)
  {
    pointer_[i] = str[i];
  }
  capacity_ = (size_ * sizeof(char));
};

rebdev::String::String(String&& str) noexcept:
  size_(0),
  capacity_(0),
  pointer_(nullptr)
{
  while (str[size_] != '\0')
  {
    size_ += 1;
  }

  pointer_ = new char[size_];
  for (size_t i = 0; i < size_; ++i)
  {
    pointer_[i] = str[i];
  }
  capacity_ = (size_ * sizeof(char));
};

rebdev::String::~String ()
{
  delete[] pointer_;
};


rebdev::String& rebdev::String::operator= (const char* str)
{
  if (pointer_ != nullptr)
  {
    delete[] pointer_;
  }

  size_ = 0;
```

```
  while (str[size_] != '\0')
  {
    size_ += 1;
  }

  pointer_ = new char[size_];
  for (size_t i = 0; i < size_; ++i)
  {
    pointer_[i] = str[i];
  }
  capacity_ = (size_ * sizeof(char));
  return *this;
};

rebdev::String& rebdev::String::operator= (String&& str) noexcept
{
  size_ = 0;
  while (str[size_] != '\0')
  {
    size_ += 1;
  }

  pointer_ = new char[size_];
  for (size_t i = 0; i < size_; ++i)
  {
    pointer_[i] = str[i];
  }
  capacity_ = (size_ * sizeof(char));
  return *this;
};

char& rebdev::String::operator[] (size_t pos) const
{
  if ((pos >= 0) && (pos < size_))
  {
    return pointer_[pos];
  }
  return pointer_[size_ - 1];
};

rebdev::String& rebdev::String::append (const String& str)
{
  return (*this + str);
};

rebdev::String& rebdev::String::operator+ (const String& str)
{
  size_t size2 = 0;
  while (str[size2] != '\0')
  {
    size2 += 1;
  }
  char * pointer = new char[size_ + size2];
  for (size_t i = 0; i < size_; ++ i)
```

```
  {
    pointer[i] = pointer_[i];
  }
  for (size_t i = 0; i < size2; ++i)
  {
    pointer[size_ + i] = str[i];
  }
  delete[] pointer_;
  pointer_ = pointer;
  pointer = nullptr;
  return *this;
};

rebdev::String& rebdev::String::insert (size_t pos, const char* str)
{
  size_t size2 = 0;
  while (str[size2] != '\0')
  {
    size2 += 1;
  }
  char * pointer = new char[size_ + size2];
  for (size_t i = 0; i < pos; ++ i)
  {
    pointer[i] = pointer_[i];
  }
  for (size_t i = 0; i < size2; ++i)
  {
    pointer[pos + i] = str[i];
  }
  for (size_t i = pos; i < size_; ++i)
  {
    pointer[pos + size2 + i] = pointer_[i];
  }
  delete[] pointer_;
  pointer_ = pointer;
  pointer = nullptr;

  return *this;
};
int rebdev::String::compare (const String& str)
{
  size_t size2 = 0;
  while (str[size2] != '\0')
  {
    size2 += 1;
  }
  if (size2 != size_)
  {
    return ((size_ > size2)? 1: -1);
  }
  for (size_t i = 0; i < size_; ++i)
  {
    if (pointer_[i] != str[i])
    {
```

```
      return ((pointer_[i] > str[i])? 1: -1);
    }
  }
  return 0;
};
```