**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра САПР**

**ОТЧЕТ**

**по лабораторной работе №4**

**по дисциплине «Программирование»**

**Тема: Операции со списками**

Студент гр. 2302 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Николаев В.Ю.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Калмычков В.А.

Санкт-Петербург

2023 г.

# Исходная формулировка

Разработать процедуру или функцию, обеспечивающую выполнение произвольного, задаваемого с помощью входного параметра набора действий для каждого элемента списка 1.

# **Организация UI**

| Макет файла input.txt:  s\_s  s\_s  ⋮  s\_s | Макет файла output.txt:  [s\_s]=>[s\_s]=>⋯=>NULL  | |  \/  [s\_s]=>[s\_s]=>⋯=>NULL  | |  \/  ⋮  [s\_s]=>[s\_s]=>⋯=>NULL  | |  \/  NULL  [s\_s]=>[s\_s]=>⋯=>NULL  | |  \/  [s\_s]=>[s\_s]=>⋯=>NULL  | |  \/  ⋮  [s\_s]=>[s\_s]=>⋯=>NULL  | |  \/  NULL |
| --- | --- |
| Макет O1:  Choose an action:\n  \t1. Add line;\n  \t2. Delete line;\n  \t3. Change line.\n  Макет O2:  There are n\_n lines in the text.\n  Макет O3:  Enter the line number after which you want to add:\n  Макет O4:  Enter the line you want to add:\n  Макет O5:  Enter the line number you want to delete:\n  Макет O6:  Enter the line number you want to change:\n  Макет O7:  Enter the line you want to change to:\n  Макет O8:  Enter a number from n\_n to n\_n .\n  Макет O9:  Enter the number.\n |
| Макет I1:  n\_n |

# Особенности системы программирования

Проект собран на g++ (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0. Для сборки используется GNU Make 4.3 Built for x86\_64-pc-linux-gnu.

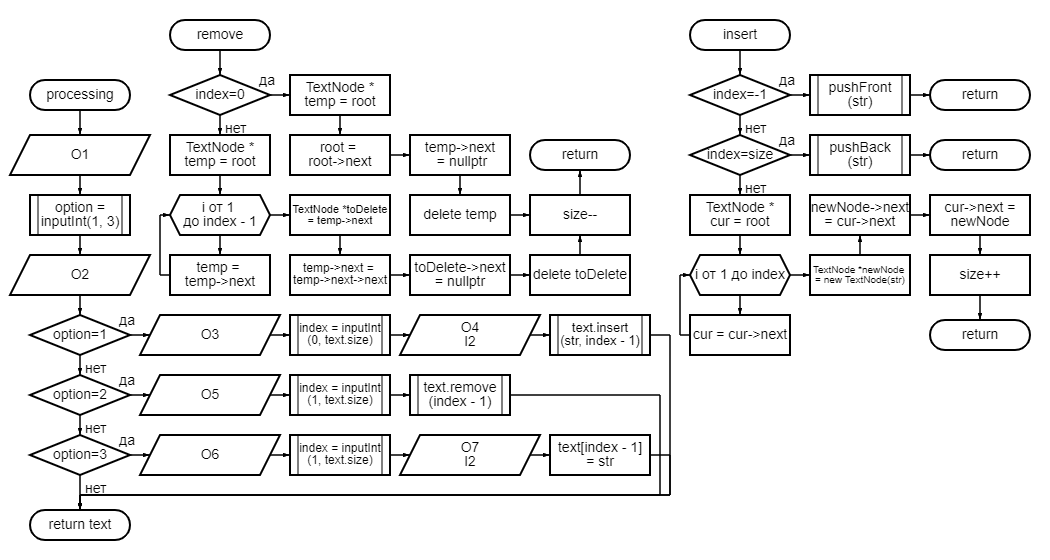
# Структуры данных

| String  class String  {  private:  StringNode \*root = nullptr;  public:  int size = 0;  public:  String();  String(const String &);  ~String();  void pushBack(char);  void pushBack(char \*);  void pushBack(String);  void clear();  bool isInt();  int toInt();  friend bool operator==(const String &, const String &);  friend std::ostream &operator<<(std::ostream &, const String &);  friend std::istream &operator>>(std::istream &, String &);  char &operator[](const int) const;  String operator=(const String &);  };  StringNode \*root = nullptr - указатель на начало строки  int size = 0; - длина строки | StringNode  class StringNode  {  public:  char \*data = nullptr;  static const int l = 10;  int size = 0;  StringNode \*next = nullptr;  public:  StringNode();  StringNode(const StringNode &);  ~StringNode();  StringNode operator=(const StringNode &);  };  char \*data = nullptr - указатель на массив символов  char \*data = nullptr - максимальное количество символом в строке  int size = 0 - количество символов в строке  StringNode \*next = nullptr - следующая часть строки |
| --- | --- |
|
|
| Text  class Text  {  private:  TextNode \*root = nullptr;  public:  int size = 0;  public:  Text();  Text(const Text &);  ~Text();    void pushBack(String);    void pushFront(String);  void insert(String, int);  void remove(int);    String &operator[](int) const;  Text operator=(const Text &);  };  TextNode \*root = nullptr - указатель на начало текста  int size = 0 - длина текста |
| TextNode  class TextNode  {  public:  String data;  TextNode \*next = nullptr;  public:  TextNode();  TextNode(String);  TextNode(const TextNode &);  ~TextNode();  TextNode operator=(const TextNode &);  };  String data - строка  TextNode \*next = nullptr - следующая часть текста |
|
|

# Описание функций

| Тип | Название | Передаваемые переменные | Описание переменных | Описание функции |
| --- | --- | --- | --- | --- |
| int | inputInt | int min | минимум | Ввод с клавиатуры переменной типа int лежащей в [min, max] |
| int max | максимум |
| Text | readFromFile | const char \*fileName | имя файла | Чтение данных из файла |
| Text | writeToFile | const char \*fileName | имя файла | Вывод данных в файл |
| Text text | текст |
| std::ios\_base::openmode openmode | режим записи |
| Text | processing | Text text | текст | Обработка текста |

# **Представление алгоритма решения задачи**



# Текст программы

| src/   | Main.cpp | | --- | | #include "../lib/Task/Task.hpp"  int main()  {  writeToFile  (  "files/output.txt",  processing  (  writeToFile  (  "files/output.txt",  readFromFile  (  "files/input.txt"  ),  std::ios::out  )  ),  std::ios::app  );  return 0;  } |   lib/   | Header.hpp | | --- | | #pragma once  #include <iostream>  #include <fstream>  #include <cassert> |   lib/StringNode/   | Operator.cpp | | --- | | #include "StringNode.hpp"  StringNode StringNode::operator=(const StringNode &other)  {  size = other.size;  data = new char[l + 1];  for (int i = 0; i < size; i++)  data[i] = other.data[i];    for (int i = size; i < l; i++)  data[i] = '0';  if (other.next != nullptr)  next = new StringNode(\*other.next);  return \*this;  } | | StringNode.cpp | | #include "StringNode.hpp"  StringNode::StringNode()  {  data = new char[l + 1];  for (int i = 0; i < l; i++)  data[i] = '0';  }  StringNode::StringNode(const StringNode &node)  {  size = node.size;  data = new char[l + 1];  for (int i = 0; i < size; i++)  data[i] = node.data[i];  for (int i = size; i < l; i++)  data[i] = '0';    if (node.next != nullptr)  next = new StringNode(\*node.next);  }  StringNode::~StringNode()  {  delete next;  next = nullptr;  delete[] data;  data = nullptr;  } | | StringNode.hpp | | #pragma once  #include "../Header.hpp"  class StringNode  {  public:  char \*data = nullptr;  static const int l = 10;  int size = 0;  StringNode \*next = nullptr;  public:  StringNode();  StringNode(const StringNode &);  ~StringNode();  StringNode operator=(const StringNode &);  }; |   lib/String/   | Clear.cpp | | --- | | #include "String.hpp"  void String::clear()  {  delete root;  root = nullptr;  size = 0;  } | | IsInt.cpp | | #include "String.hpp"  bool String::isInt()  {  if (size == 0)  return false;    StringNode \*cur = root;    int i = 0;  if (cur->data[0] == '-' || cur->data[0] == '+')  i++;    for (; i < size; i++)  {  char c = cur->data[i % cur->l];  if (c < '0' || c > '9')  return false;  }    return true;  } | | Operator.cpp | | #include "String.hpp"  bool operator==(const String &left, const String &right)  {  if (left.size == right.size)  {  StringNode \*lcur = left.root;  StringNode \*rcur = right.root;    while (lcur != nullptr)  if (lcur != rcur)  return false;  else  {  lcur = lcur->next;  rcur = rcur->next;  }  }    return left.size == right.size;  }  std::ostream &operator<<(std::ostream &os, const String &str)  {  StringNode \*cur = str.root;  while (cur != nullptr)  {  os << "[";  for (int i = 0; i < cur->size; i++)  if (cur->data[i] != '\r')  os << cur->data[i];  os << "]=>";    cur = cur->next;  }  os << "NULL\n";  return os;  }  std::istream &operator>>(std::istream &is, String &str)  {  is.unsetf(std::ios::skipws);  char c;  while(is >> c && c != '\n')  str.pushBack(c);    return is;  }  char &String::operator[](const int n) const  {  assert((void("E1"), 0 <= n));  assert((void("E1"), n < size));  StringNode \*cur = root;  int i = n;  for (; i > root->l; i -= root->l)  cur = cur->next;  return cur->data[i];  }  String String::operator=(const String &str)  {  size = str.size;  if (str.root != nullptr)  root = new StringNode(\*str.root);  return \*this;  } | | PushBack.cpp | | #include "String.hpp"  void String::pushBack(char c)  {  size++;  if (root == nullptr)  {  root = new StringNode();  root->data[root->size++] = c;  return;  }  StringNode \*cur = root;  while (cur->next != nullptr)  cur = cur->next;    if (cur->size == root->l)  {  cur->next = new StringNode();  cur = cur->next;  cur->data[cur->size++] = c;  return;  }    cur->data[cur->size++] = c;  }  void String::pushBack(char \*str)  {  for (int i = 0; int(str[i]) != '\0'; i++)  pushBack(str[i]);  }  void String::pushBack(String str)  {  for (int i = 0; i < str.size; i++)  pushBack(str[i]);  } | | String.cpp | | #include "String.hpp"  String::String() {}  String::String(const String &str)  {  size = str.size;  if (str.root != nullptr)  root = new StringNode(\*str.root);  }  String::~String()  {  delete root;  root = nullptr;  } | | String.hpp | | #pragma once  #include "../StringNode/StringNode.hpp"  class String  {  private:  StringNode \*root = nullptr;  public:  int size = 0;  public:  String();  String(const String &);  ~String();  void pushBack(char);  void pushBack(char \*);  void pushBack(String);  void clear();  bool isInt();  int toInt();  friend bool operator==(const String &, const String &);  friend std::ostream &operator<<(std::ostream &, const String &);  friend std::istream &operator>>(std::istream &, String &);  char &operator[](const int) const;  String operator=(const String &);  }; | | ToInt.cpp | | #include "String.hpp"  int String::toInt()  {  assert((void("E6"), size == 0 or isInt()));  StringNode \*cur = root;  int res = 0;  int sign = 1;  int i = 0;  if (cur->data[0] == '-')  {  sign = -1;  i++;  }  else if (cur->data[0] == '+')  i++;  for (; i < size; i++)  {  res \*= 10;  res += cur->data[i % cur->l] - '0';  }  return res \* sign;  } |   lib/Task/   | Task.hpp | | --- | | #pragma once  #include "../Text/Text.hpp"  int inputInt(int, int);  Text readFromFile(const char \*);  Text writeToFile(const char \*, Text, std::ios\_base::openmode);  Text processing(Text); | | lib/TextNode/   | Operator.cpp | | --- | | #include "TextNode.hpp"  TextNode TextNode::operator=(const TextNode &node)  {  data = node.data;  if (node.next != nullptr)  next = new TextNode(\*node.next);  return \*this;  } | | TextNode.cpp | | #include "TextNode.hpp"  TextNode::TextNode() {}  TextNode::TextNode(String str)  {  data = str;  }  TextNode::TextNode(const TextNode &node)  {  data = node.data;  if (node.next != nullptr)  next = new TextNode(\*node.next);  }  TextNode::~TextNode()  {  delete next;  next = nullptr;  } | | TextNode.hpp | | #pragma once  #include "../String/String.hpp"  class TextNode  {  public:  String data;  TextNode \*next = nullptr;  public:  TextNode();  TextNode(String);  TextNode(const TextNode &);  ~TextNode();  TextNode operator=(const TextNode &);  }; |   lib/Text/   | Insert.cpp | | --- | | #include "Text.hpp"  void Text::insert(String str, int index)  {  assert((void("E7"), -1 <= index and index < size));  if (index == -1)  {  pushFront(str);  return;  }  if (index == size)  {  pushBack(str);  return;  }  TextNode \*cur = root;  for (int i = 0; i < index; i++)  cur = cur->next;  TextNode \*newNode = new TextNode(str);  newNode->next = cur->next;  cur->next = newNode;  size++;  } | | Operator.cpp | | #include "Text.hpp"  String &Text::operator[](int n) const  {  assert((void("E2"), n < size));  assert((void("E2"), 0 <= n));    TextNode \*cur = root;  int i = n;  while (i--)  {  cur = cur->next;  assert((void("E5"), cur != nullptr));  }  return cur->data;  }  Text Text::operator=(const Text &text)  {  size = text.size;  if (text.root != nullptr)  root = new TextNode(\*text.root);  return \*this;  } | | PushBack.cpp | | #include "Text.hpp"  void Text::pushBack(String str)  {  size++;  if (root == nullptr)  {  root = new TextNode(str);  return;  }    TextNode \*cur = root;  while (cur->next != nullptr)  cur = cur->next;    cur->next = new TextNode(str);  } | | PushFront.cpp | | #include "Text.hpp"  void Text::pushFront(String str)  {  TextNode \*newNode = new TextNode(str);  newNode->next = root;  root = newNode;  size++;  } | | Remove.cpp | | #include "Text.hpp"  void Text::remove(int index)  {  assert((void("E8"), 0 <= index and index < size));  if (index == 0)  {  TextNode \*temp = root;  root = root->next;  temp->next = nullptr;  delete temp;  }  else  {  TextNode \*temp = root;  for (int i = 0; i < index - 1; i++)  temp = temp->next;  TextNode \*toDelete = temp->next;  temp->next = temp->next->next;  toDelete->next = nullptr;  delete toDelete;  }  size--;  } | | Text.cpp | | #include "Text.hpp"  Text::Text()  {}  Text::Text(const Text &text)  {  size = text.size;  if (text.root != nullptr)  root = new TextNode(\*text.root);  }  Text::~Text()  {  delete root;  root = nullptr;  } | | Text.hpp | | #pragma once  #include "../TextNode/TextNode.hpp"  class Text  {  private:  TextNode \*root = nullptr;  public:  int size = 0;  public:  Text();  Text(const Text &);  ~Text();    void pushBack(String);    void pushFront(String);  void insert(String, int);  void remove(int);    String &operator[](int) const;  Text operator=(const Text &);  }; |   lib/Task/   | InputInt.cpp | | --- | | #include "Task.hpp"  int inputInt(int min, int max)  {  int n = -10;  String str;  while (n < min || max < n)  {  std::cin >> str;  if (str.isInt())  {  n = str.toInt();    if (n < min || max < n)  std::cout << "Enter a number from " << min << " to " << max << ".\n";  }  else  std::cout << "Enter the number.\n";  str.clear();  }  return n;  } | | Processing.cpp | | #include "Task.hpp"  Text processing(Text text)  {  std::cout << "Choose an action:\n"  << "\t1. Add line;\n"  << "\t2. Delete line;\n"  << "\t3. Change line.\n";  int option = inputInt(1, 3);  int index;  String str;  std::cout << "There are " << text.size << " lines in the text.\n";  switch (option)  {  case 1:  std::cout << "Enter the line number after which you want to add:\n";  index = inputInt(0, text.size);  std::cout << "Enter the line you want to add:\n";  std::cin >> str;  text.insert(str, index - 1);  break;    case 2:  std::cout << "Enter the line number you want to delete:\n";  index = inputInt(1, text.size);  text.remove(index - 1);  break;    case 3:  std::cout << "Enter the line number you want to change:\n";  index = inputInt(1, text.size);  std::cout << "Enter the line you want to change to:\n";  std::cin >> str;  text[index - 1] = str;  break;  }  return text;  } | | ReadFromFile.cpp | | #include "Task.hpp"  Text readFromFile(const char \*fileName)  {  std::fstream fin(fileName, std::ios::in);  assert((void("E3"), fin.is\_open()));  fin << std::noskipws;  Text text;  String str;  while (fin >> str)  {  text.pushBack(str);  str.clear();  }  if (str.size != 0)  text.pushBack(str);  fin.close();  return text;  } | | WriteToFile.cpp | | #include "Task.hpp"  Text writeToFile(const char \*fileName, Text text, std::ios\_base::openmode openmode)  {  std::fstream fout(fileName, openmode);  assert((void("E4"), fout.is\_open()));  if (openmode == std::ios::app)  fout << "\n";  for (int i = 0; i < text.size; i++)  fout << text[i] << "||\n\\/\n";  fout << "NULL\n";  fout.close();  return text;  } | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

# Выводы о проделанной работе

В этой лабораторной работе мы научились производить операции со списками.