Министерство науки и образования Российской Федерации

Федеральное государственное бюджетное образовательноеучреждение

высшего образования

«Волгоградский государственный технический университет»

КАЧЕСТВО И НАДЕЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Программа для построения связей в генеалогическом древеВнешняя спецификация

СОГЛАСОВАНО:

Руководитель проекта:

Доцент кафедры ПОАС

_____Сычев О. А.

«___»_____2021 г.
г.

Разработчик:

Студент группы

ПрИн-266

_____Козарез М.В.

«___»_____2021

Нормоконтролер:

Преподаватель кафедры ПОАС

_____Матюшечкин Д.С.

«___»_____2021
г.

Волгоград 2021 г.

Оглавление

1 Проектирование функции translit, решающей главную задачу

1.1 Модульные тесты

1.1.1 Реализация модульных тестов

```
TEST_CLASS(TestLibrary)
	{
	public:


		TEST_METHOD(_translitMin)
		{
			setDecMap();
			string eng = "q";
			vector<string> exp_string{ "ку" };

			vector<string> returnString = translit(eng);
			if (exp_string == returnString)
			{
				Assert::AreEqual(1, 1);
			}
			else
			{
				Assert::AreEqual(1, 2);
			}
		}


		TEST_METHOD(_translitMedium)
		{
			setDecMap();
			string eng = "qqqqq";
			vector<string> exp_string{ "кукукукуку" };

			vector<string> returnString = translit(eng);
			if (exp_string == returnString)
			{
				Assert::AreEqual(1, 1);
			}
			else
			{
				Assert::AreEqual(1, 2);
			}
		}


		TEST_METHOD(_translitMax)
		{
			setDecMap();
			string eng = "qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq";
			vector<string> exp_string{
"кукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукукуку
кукукукукукуку" };

			vector<string> returnString = translit(eng);
			if (exp_string == returnString)
			{
				Assert::AreEqual(1, 1);
			}
			else
			{
				Assert::AreEqual(1, 2);
			}
		}
```

```
59
60
61          TEST_METHOD(_multipleTranslitTwoVariants)
62          {
63                  setDecMap();
64                  string eng = "ezhik";
65                  vector<string> exp_string{ "езхик", "ежик" };
66
67                  vector<string> returnString = translit(eng);
68                  if (exp_string == returnString)
69                  {
70                          Assert::AreEqual(1, 1);
71                  }
72                  else
73                  {
74                          Assert::AreEqual(1, 2);
75                  }
76          }
77
78
79          TEST_METHOD(_multipleTranslitFourVariants)
80          {
81                  setDecMap();
82                  string eng = "yozhik";
83                  vector<string> exp_string{ "йозхик", "йожик", "ёзхик", "ёжик" };
84
85                  vector<string> returnString = translit(eng);
86                  if (exp_string == returnString)
87                  {
88                          Assert::AreEqual(1, 1);
89                  }
90                  else
91                  {
92                          Assert::AreEqual(1, 2);
93                  }
          }
```

1.2 Код

```
vector<string> translit(string eng)
{
    string copy = makeLow(eng);
    decodes.resize(copy.size() + 1);
    decodes[copy.size()] = { "" };
    vector<string>ru = decodeString(copy, 0);
    return ru;
}
```

2. Проектирование функции decodeString, решающей главную задачу

2.1 Код

```
vector<string> decodeString(string& line, int pos)
{
    if (decodes[pos].size() > 0)
        return decodes[pos];
    else
    {
        vector<string>res;
        for (int len = 1; (len <= maxCombLen) && (pos + len <= line.size()); ++len)
        {
```

```
            vector<string>& curVars = decMap[line.substr(pos, len)],
                leftVars = decodeString(line, pos + len);
            if (upper[pos])
                for (int i = 0; i < curVars.size(); ++i)
                    curVars[i] = toupper(curVars[i]);
            for (int i = 0; i < curVars.size(); ++i)
                for (int j = 0; j < leftVars.size(); ++j)
                    res.push_back(curVars[i] + leftVars[j]);
        }
        decodes[pos] = res;
        return res;
    }
}
```

3. Проектирование функции tolower

3.1 Модульные тесты

3.1.1 Реализация модульных тестов

```
TEST_METHOD(_tolowerChar)
		{
			setDecMap();
			char symbol = 'K';
			char exp_symbol = 'k';

			char returnChar = tolower(symbol);
			Assert::AreEqual(exp_symbol, returnChar);
		}
```

3.2 Код

```
char tolower(char ch)
{
    return static_cast<char>(tolower(static_cast<unsigned char>(ch)));
}
```

4. Проектирование функции toupper(char)

4.1 Модульные тесты

4.1.1 Реализация модульных тестов

```
TEST_METHOD(_toupperChar)
		{
			setDecMap();
			char symbol = 'k';
			char exp_symbol = 'K';

			char returnChar = toupper(symbol);
			Assert::AreEqual(exp_symbol, returnChar);
		}
```

4.2 Код

```
char toupper(char ch)
{
    return static_cast<char>(toupper(static_cast<unsigned char>(ch)));
```

```
}
```

5. Проектирование функции makeLow

5.1 Модульные тесты

5.1.1 Реализация модульных тестов

```
TEST_METHOD(_makeLow)
		{
			setDecMap();
			string line = "YOZHIK";
			string exp_string = "yozhik";

			string returnString = makeLow(line);
			Assert::AreEqual(exp_string, returnString);
		}
```

5.2 Код

```
string makeLow(string line)
{
    upper.resize(line.size());
    for (int i = 0; i < line.size(); ++i)
    {
        char c = tolower(line[i]);
        upper[i] = (c != line[i]);
        line[i] = c;
    }
    return line;
}
```

6. Проектирование функции toupper(string)

6.1 Модульные тесты

6.1.1 Реализация модульных тестов

```
TEST_METHOD(_toupperString)
		{
			setDecMap();
			string line = "yozhik";
			string exp_string = "YOZHIK";

			string returnString = toupper(line);
			Assert::AreEqual(exp_string, returnString);
		}
```

6.2 Код

```
string toupper(string line)
{
    for (int i = 0; i < line.size(); ++i)
        line[i] = toupper(line[i]);
    return line;
}
```

7. Проектирование функции inputFile

7.1 Код

```
tring inputFile(string fileName)
{
    ifstream fin;

    fin.open(fileName);
    if (!fin.is_open())
    {
        printf("Ошибка при открытии файла!\n");
        exit(EXIT_FAILURE);
    }

    string line;

    int count = 0;
    while (!fin.eof())
    {
        getline(fin, line);
        count++;
    }
    if (count > 1)
    {
        printf("Количество строк в файле не равно 1!\n");
        exit(EXIT_FAILURE);
    }

    for (int i = 0; i < line.length(); i++)
    {
        if (!((line[i] >= 'A' && line[i] <= 'Z') || (line[i] >= 'a' && line[i] <= 'z') ||
line[i] == ' '))
        {
            printf("В исходной строке есть недопустимые символы!\n");
            exit(EXIT_FAILURE);
        }
    }

    cout << line << endl;

    fin.close();
    return line;
}
```

8. Проектирование функции setDecMap

8.1 Код

```
map<string, vector<string>>decMap;


void setDecMap() {
    decMap["a"] = { "а" };
    decMap["b"] = { "б" };
    decMap["c"] = { "ц" };
    decMap["d"] = { "д" };
    decMap["e"] = { "е" };
    decMap["f"] = { "ф" };
    decMap["g"] = { "г" };
    decMap["h"] = { "х" };
    decMap["i"] = { "и" };
```

```
decMap["j"] = { "дж" };
decMap["k"] = { "к" };
decMap["l"] = { "л" };
decMap["m"] = { "м" };
decMap["n"] = { "н" };
decMap["o"] = { "о" };
decMap["p"] = { "п" };
decMap["q"] = { "ку" };
decMap["r"] = { "р" };
decMap["s"] = { "с" };
decMap["t"] = { "т" };
decMap["u"] = { "у" };
decMap["v"] = { "в" };
decMap["w"] = { "в" };
decMap["x"] = { "кс" };
decMap["y"] = { "й" };
decMap["z"] = { "з" };
decMap["yo"] = { "ё" };
decMap["yu"] = { "ю" };
decMap["ya"] = { "я" };
decMap["ye"] = { "е" };
decMap["zh"] = { "ж" };
decMap["sh"] = { "ж" };
decMap["kh"] = { "х" };
decMap["eh"] = { "э" };
decMap["ih"] = { "ы" };
decMap["ts"] = { "ц" };
decMap["ch"] = { "ч" };
decMap["A"] = { "А" };
decMap["B"] = { "Б" };
decMap["C"] = { "Ц" };
decMap["D"] = { "Д" };
decMap["E"] = { "Е" };
decMap["F"] = { "Ф" };
decMap["G"] = { "Г" };
decMap["H"] = { "Х" };
decMap["I"] = { "И" };
decMap["J"] = { "Дж" };
decMap["K"] = { "К" };
decMap["L"] = { "Л" };
decMap["M"] = { "М" };
decMap["N"] = { "Н" };
decMap["O"] = { "О" };
decMap["P"] = { "П" };
decMap["Q"] = { "Ку" };
decMap["R"] = { "Р" };
decMap["S"] = { "С" };
decMap["T"] = { "Т" };
decMap["U"] = { "У" };
decMap["V"] = { "В" };
decMap["W"] = { "В" };
decMap["X"] = { "Кс" };
decMap["Y"] = { "Й" };
decMap["Z"] = { "З" };
decMap["Yo"] = { "Ё" };
decMap["Yu"] = { "Ю" };
decMap["Ya"] = { "Я" };
decMap["Ye"] = { "Е" };
decMap["Zh"] = { "Ж" };
decMap["Sh"] = { "Ш" };
decMap["Kh"] = { "Х" };
decMap["Eh"] = { "Э" };
decMap["Ih"] = { "Ы" };
decMap["Ts"] = { "Ц" };
```

```
    decMap["Ch"] = { "Ч" };
    decMap[" "] = { " " };
}
```

9. Проектирование функции outputFile

9.1 Код

```
void outputFile(vector<string> ru, string outputFileName)
{
    ofstream fout;

    fout.open(outputFileName);
    if (!fout.is_open())
    {
        printf("Ошибка при открытии файла!\n");
        exit(EXIT_FAILURE);
    }
    for (int i = 0; i < ru.size(); ++i)
        fout << ru[i] << endl;

    fout.close();
}
```