

## Спецификация лабораторной работы № 4

В данной программе создается две функции, которые работают с строками в стиле C и с типом string C++. Строки в стиле C помещены в динамическую память. Входные данные поступают из двух файлов input.txt (для строк в стиле C) и input2.txt (для строк типа string C++). Вывод новой строки происходит в файл output.txt.

Задание:

- 5) Сформировать новую строку из символов двух исходных строк, которые НЕ являются для них общими.

### Тестовый набор с выходными данными

В input.txt и input2.txt предложены варианты строк с их размерами.

```
input.txt  input2.txt  Labwork 4.cpp  output.txt
1  11
2  hello world
3  7
4  labwork
5
```

```
input.txt  input2.txt  Labwork 4.cpp  output.txt
1  hello world
2  labwork
```

Вывод:

```
input.txt  input2.txt  Labwork 4.cpp  output.txt
1  he dabk
2  he dabk
```

### Программа:

```
#include <iostream>
#include <string>
#include <cctype>

#pragma warning(disable : 4996)

std::string stringFunction(std::string source, std::string source_2);
char* cstringFunction(char* destination, char* source, char* source_2, int size, int size_2, int count);
void countFunction(const char* pcStringOne, const char* pcStringTwo, int size, int size_2, int& count);
```

```

int main()
{
    setlocale(LC_ALL, "ru");

    freopen("input.txt", "r", stdin);

    int sizeOne;
    int sizeTwo;
    int count = 0;

    std::cin >> sizeOne;
    sizeOne++;
    char* pcStringOne = new char[sizeOne];
    getc(stdin); // getc() - функция возвращает символ из потока ввода.
    gets_s(pcStringOne, sizeOne); //gets_s() - функция считывает символы из стандартного потока
    ввода до символа новой строки.

    std::cin >> sizeTwo;
    sizeTwo++;
    char* pcStringTwo = new char[sizeTwo];
    getc(stdin);
    gets_s(pcStringTwo, sizeTwo);

    countFunction(pcStringOne, pcStringTwo, sizeOne, sizeTwo, count);
    char* result = new char[count];
    result = cstringFunction(result, pcStringOne, pcStringTwo, sizeOne, sizeTwo, count);
    fclose(stdin);

    freopen("input2.txt", "r", stdin);
    std::string pcOne;
    std::string pcTwo;
    std::string resultString;

    std::getline(std::cin, pcOne); //getline() - функция для ввода данных из потока с типом
string
    std::getline(std::cin, pcTwo);
    resultString = stringFunction(pcOne, pcTwo);
    fclose(stdin);

    freopen("output.txt", "w", stdout);

    for (int i = 0; i < count; i++)
    {
        std::cout << result[i];
    }

    std::cout << '\n' << resultString;

    fclose(stdout);

    delete[] pcStringOne;
    delete[] pcStringTwo;
    return 0;
}

char* cstringFunction(char* destination, char* source, char* source_2, int size, int size_2, int
count)
{
    int temp = 0;
    for (int i = 0; i < size; i++)
    {
        bool flag = true;
        for (int j = 0; j < size_2; j++)
        {
            if (source[i] == source_2[j])
            {
                flag = false;
            }
        }
    }
}

```

```

    }
    if (flag)
    {
        destination[temp] = source[i];
        temp++;
    }
}
for (int i = 0; i < size_2; i++)
{
    bool flag = true;
    for (int j = 0; j < size; j++)
    {
        if (source_2[i] == source[j])
        {
            flag = false;
        }
    }
    if (flag)
    {
        destination[temp] = source_2[i];
        temp++;
    }
}
return destination;
}

```

```

std::string stringFunction(std::string source, std::string source_2)
{
    std::string destination;
    int temp = 0;
    for (int i = 0; i < source.length(); i++)
    {
        bool flag = true;
        for (int j = 0; j < source_2.length(); j++)
        {
            if (source[i] == source_2[j])
            {
                flag = false;
            }
        }
        if (flag)
        {
            destination.push_back(source[i]); //push_back() - функция, добавляющая ячейку в
стек(обычно используется с vector).
            temp++;
        }
    }
    for (int i = 0; i < source_2.length(); i++)
    {
        bool flag = true;
        for (int j = 0; j < source.length(); j++)
        {
            if (source_2[i] == source[j])
            {
                flag = false;
            }
        }
        if (flag)
        {
            destination.push_back(source_2[i]);
            temp++;
        }
    }
    return destination;
}

```

```
void countFunction(const char* pcStringOne, const char* pcStringTwo, int size, int size_2, int&
count)
{
    for (int i = 0; i < size; i++)
    {
        bool flag = true;
        for (int j = 0; j < size_2; j++)
        {
            if (pcStringOne[i] == pcStringTwo[j])
            {
                flag = false;
            }
        }
        if (flag)
        {
            count++;
        }
    }
    for (int i = 0; i < size_2; i++)
    {
        bool flag = true;
        for (int j = 0; j < size; j++)
        {
            if (pcStringTwo[i] == pcStringOne[j])
            {
                flag = false;
            }
        }
        if (flag)
        {
            count++;
        }
    }
}
```