KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Programavimo kalbų teorija (P175B124)

Laboratorinių darbų ataskaita

Atliko:
IFF-1/9 gr. studentas
Nedas Liaudanskis
2024 m. birželio 20 d.
Priėmė:
Lekt. Guogis Evaldas

KAUNAS 2023

TURINYS

C++ arba Ruby(L1)

1.1. Darbo užduotis

619 – Numerically Speaking

A developer of crossword puzzles (and other similar word games) has decided to develop a mapping between every possible word with from one to twenty characters and unique integers. The mapping is very simple, with the ordering being done first by the length of the word, and then alphabetically. Part of the list is shown below.

```
a             1
b             2
...
z             26
aa            27
ab            28
...
snowfall      157,118,051,752
...
```

Your job in this problem is to develop a program which can translate, bidirectionally, between the unique word numbers and the corresponding words.

Input

Input to the program is a list of words and numbers, one per line starting in column one, followed by a line containing a single asterisk in column one. A number will consist only of decimal digits (0 through 9) followed immediately by the end of line (that is, there will be no commas in input numbers). A word will consist of between one and twenty lowercase alphabetic characters (a through z).

Output

The output is to contain a single line for each word or number in the input data. This line is to contain the word starting in column one, followed by an appropriate number of blanks, and the corresponding word number starting in column 23. Word numbers that have more than three digits must be separated by commas at thousands, millions, and so forth.

Sample Input

```
29697684282993
transcendental
28011622636823854456520
computationally
zzzzzzzzzzzzzzzzzzz
*
```

Sample Output

```
elementary              29,697,684,282,993
transcendental          51,346,529,199,396,181,750
prestidigitation        28,011,622,636,823,854,456,520
computationally         232,049,592,627,851,629,097
zzzzzzzzzzzzzzzzzzz     20,725,274,851,017,785,518,433,805,270
```

Užduoties pilnai įvykdyti nepavyko, nes nesugebėjau perskaityti kintamųjų arba su jais atlikti skaičiavimų, kurie buvo didesni nei unsigned long long int tipo kintamasis. Todėl skaitydamas failus uždėjau skaičių limitą ir raidžių limitą (19 skaičių ir 13 raidžių). Tikiuosi tai nėra labai didelė problema.

1.2. Programos tekstas:

```
// Lab1.cpp : This file contains the 'main' function. Program execution begins and ends
there.
//

#include <iostream>
#include <fstream>
#include <list>
#include <iterator>
#include <array>
#include <iterator>
#include <string>
#include <vector>
#include <chrono>
#include <sstream>
using namespace std;

/// <summary>
/// An object Class that defines the word, its numerical meaning and holds the output
string.
/// </summary>
class NumericalData
{
private:

    /// <summary>
    /// String type variable that has the word
    /// </summary>
    string Word;
    /// <summary>
    /// Int type variable that holds the numarical form of the word
    /// </summary>
    unsigned long long int Number;
    /// <summary>
    /// Output string
    /// </summary>
    string OutPut;

    /// <summary>
    /// Function that Converts the given word into numbers using the method given in the
exercise
    void StringToNumbers()
    {
        unsigned long long int Numb = 0;
        unsigned long long int x = 1;

        for (int i = Word.length() - 1; i >= 0; i--)
        {
            Numb += (Word[i] - 'a' + 1) * x;
            x = x * 26;
        }
        this->Number = Numb;
    }

    /// <summary>
    /// Function that converts numbers(int type variables) into string type words, using
the method given in the exercise
    /// </summary>
    void NumbersToString()
    {
        string line = "";
        unsigned long long int Saved = Number;
        while (Number != 0)
        {
            line += (char)('a' + (Number - 1) % 26);
            this->Number = (Number - 1) / 26;
```

```
        }
        reverse(line.begin(), line.end());
        this->Word = line;
        this->Number = Saved;
    }
    /// <summary>
    /// Converts an integer into the required output string
    /// </summary>
    void ForOutPut()
    {
        string Out = to_string(Number);
        int count = 0;
        for (int i = Out.length() - 1; i >= 0; i--)
        {
            count++;
            if (count % 3 == 0 && i != 0)
            {
                Out.insert(i, ",");
            }
        }

        this->OutPut = Out;
    }

public:
    /// <summary>
    /// Constructor used for receiving only the number ar the given option
    /// </summary>
    /// <param name="Num"> long long int type value that show the number that will be
needed to decrypt </param>
    NumericalData(unsigned long long int Num)
    {
        Word = "";
        Number = Num;
        OutPut = "";
    }

    /// <summary>
    /// Constructor used for receiving string type variable as the given option (the
given word)
    /// </summary>
    /// <param name="word"> The string object that is the word that will be turned into
numerical values </param>
    NumericalData(string word)
    {

        Word = word;
        Number = 0;
        OutPut = "";
    }

    /// <summary>
    /// The method used to calculate the remaining values of all the words and numerical
numbers that are in this object
    /// </summary>
    void CalculateValues()
    {
        if (Word == "")
        {

            NumbersToString();

            ForOutPut();
        }
        else
        {

            StringToNumbers();
```

```
            ForOutPut();
        }
    }

    /// <summary>
    /// Returns the starting string (used for output)
    /// </summary>
    /// <returns> starting string(word) </returns>
    string ReturnStartingString()
    {
        return Word;
    }
    /// <summary>
    /// Returns the output string
    /// </summary>
    /// <returns> The output string </returns>
    string ReturnOutput()
    {
        return OutPut;
    }

};

/// <summary>
/// A not finished class that was used to test a way to store bigger integer numbers
/// </summary>
class Utils
{
    int* GetBiggerInteger(string str)
    {
        int x = str.size();
        int* Integer = new int[str.size()];
        int a = 0;
        while (a != x)
        {
            Integer[a] = str[a] - '0';
            a++;
        }
        return Integer;

    }
};

/// <summary>
/// A cointainer that holds the NumericalData objects
/// </summary>
class NumericalDataContainer
{
private:
    /// <summary>
    /// NumericalData object vector
    /// </summary>
    vector<NumericalData> AllData;

public:

    /// <summary>
    /// Constructor used in this class, creates an empty vector
    /// </summary>
    NumericalDataContainer()
    {
        AllData = {};
    }

    /// <summary>
    /// A method used to add data to the container
    /// </summary>
```

```
    /// <param name="data"> NumericalData object tjat needs t be added to the container
</param>
    void AddData(NumericalData data)
    {
        AllData.push_back(data);
    }

    /// <summary>
    /// Return the Numericaldata object that was stored in this containers index
    /// </summary>
    /// <param name="index"> Index of the object we want to get </param>
    /// <returns> NumericalData object that matches the given index </returns>
    NumericalData ReturnDataByIndex(int index)
    {
        return AllData.at(index);
    }

    /// <summary>
    /// Returns how many objects are stored inside the container
    /// </summary>
    /// <returns> How many objects are stored inside the container </returns>
    int Count()
    {
        return AllData.size();
    }

    /// <summary>
    /// A method used to calculate all NumericalData's missing values(results)
    /// </summary>
    void CalculateData()
    {
        int count = 0;
        for (NumericalData Data : AllData)
        {
            Data.CalculateValues();
            AllData.at(count) = Data;
            count++;
        }
    }
};

/// <summary>
/// A class used for  reading data from the ttxt file and printing results into the
console
/// </summary>
class InOut {

    public:

    /// <summary>
    /// Constructor
    /// </summary>
    InOut() {

    };

    /// <summary>
    /// A static method that returns a container full of NumericalData objects, that are
read from the txt file
    /// </summary>
    /// <param name="File"> Name of the file where the data is beeing stored at </param>
    /// <returns> NumericalDataContainer object that is a container for all NumericalData
objects in this project </returns>
    static NumericalDataContainer ReadFile(string File)
    {
        NumericalDataContainer AllData;
        fstream myFile;
        myFile.open(File);
```

```
        if (myFile.is_open())
        {

            string line;
            while (getline(myFile, line))
            {



                if (isdigit(line[1]))
                {
                    if (line.length() <= 19)
                    {
                        unsigned long long int digit = stoll(line);

                        NumericalData Data(digit);
                        AllData.AddData(Data);
                    }
                    else
                    {
                        cout << "Number: " << line << " Was longer then the max length of
19 digits, so the calculations couldn't be done." << endl;
                    }

                }
                else
                {
                    if (line.length() <= 13)
                    {
                        NumericalData Data(line);

                        AllData.AddData(Data);

                    }
                    else
                    {
                        cout << "Line: " << line << " Was longer then the max length of a
string type objects (13 letters), so calculations couldn't be done." << endl;
                    }

                }

            }
        }
        myFile.close();
        return AllData;
    }



    /// <summary>
    /// Print the fial answers into the console window
    /// </summary>
    /// <param name="AllData"> Container storing all the data </param>
    static void PrintOutPut(NumericalDataContainer AllData)
    {
        for (int i = 0; AllData.Count() > i; i++)
        {
            cout << "String: " << AllData.ReturnDataByIndex(i).ReturnStartingString() <<
" Number: " << AllData.ReturnDataByIndex(i).ReturnOutput() << endl;
        }
    }

};
```

```
/// <summary>
/// Main function used to run the program
/// </summary>
/// <returns></returns>
int main()
{
    /// <summary>
    /// std::chrono function used to get the time before the program runs,
    /// that will be used to calculate the time it took for the program to do all the
calculations.
    /// </summary>
    /// <returns></returns>
    auto start = std::chrono::high_resolution_clock::now();
    InOut Out;
    NumericalDataContainer Data = Out.ReadFile("Duom.txt");
    Data.CalculateData();
    Out.PrintOutPut(Data);
    auto stop = std::chrono::high_resolution_clock::now();
    auto time = std::chrono::duration_cast<std::chrono::seconds>(stop - start);

    cout << "Time taken: " << time.count();
}
// Run program: Ctrl + F5 or Debug > Start Without Debugging menu
// Debug program: F5 or Debug > Start Debugging menu

// Tips for Getting Started:
//   1. Use the Solution Explorer window to add/manage files
//   2. Use the Team Explorer window to connect to source control
//   3. Use the Output window to see build output and other messages
//   4. Use the Error List window to view errors
//   5. Go to Project > Add New Item to create new code files, or Project > Add Existing
Item to add existing code files to the project
//   6. In the future, to open this project again, go to File > Open > Project and select
the .sln file
```

1.3. Pradiniai duomenys ir rezultatai:

Duomenys:

29697684282993
snowfall
28011622636823854456520
finally
zzzzzzzzzzzz
99246114928149462

Rezultatai:

Number: 28011622636823854456520 Was longer then the max length of 19 digits, so the calculations couldn't be done.
String: elementary Number: 29,697,684,282,993
String: snowfall Number: 157,118,051,752
String: finally Number: 1,966,850,729
String: zzzzzzzzzzzz Number: 99,246,114,928,149,462

String: zzzzzzzzzzzz Number: 99,246,114,928,149,462
Time taken: 0