Roman Vasilyev

Prof Ross

CISP 430

4/3/2024

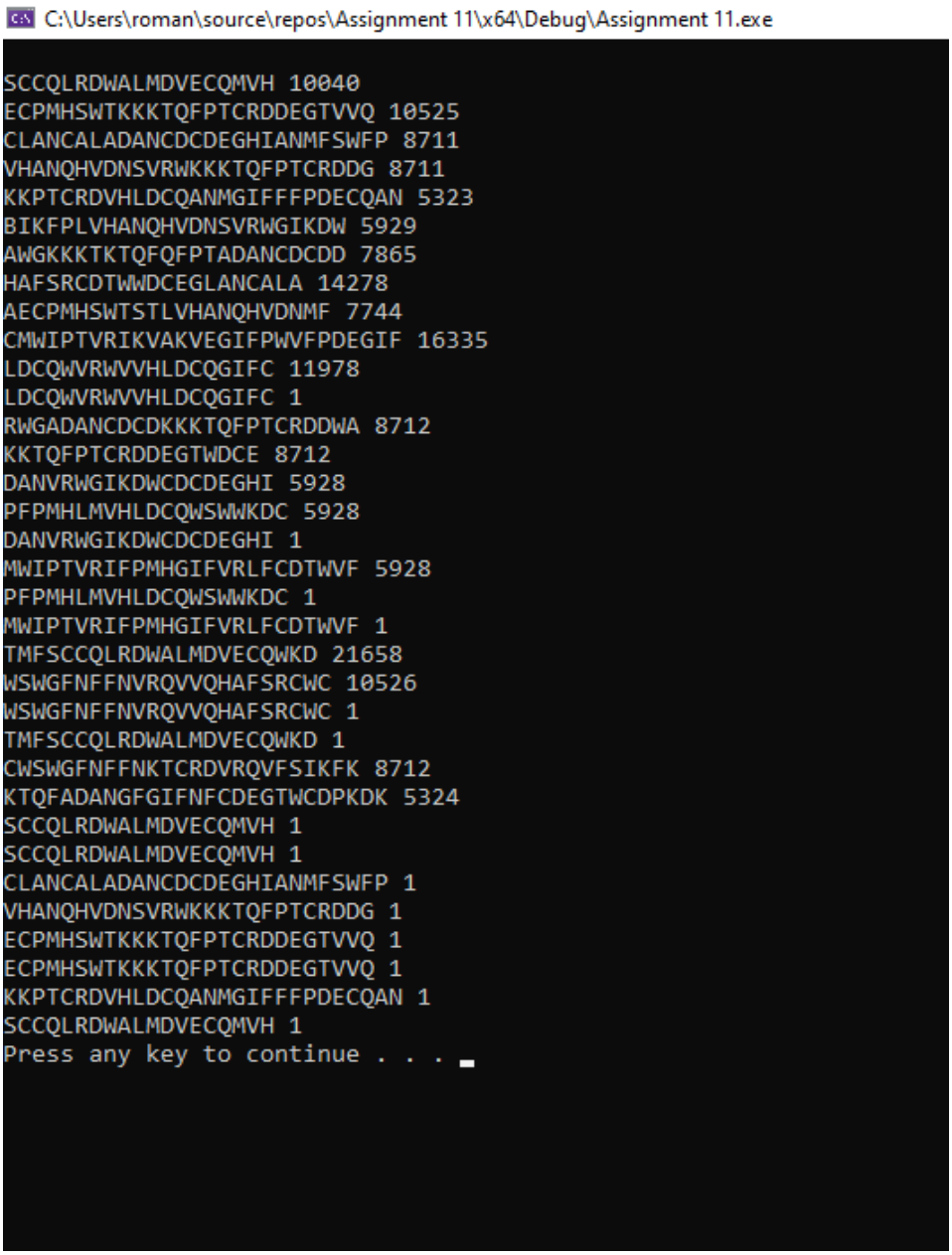
HASHING IMPLEMENTATION

//Code without user input

```
#include<iostream>
#include<string.h>
#include<cstdlib>
#include<fstream>
using namespace std;
/*structure to hold protein sequence and its count*/
struct arrayelement
{
	char protein[30];
	int count;
};
arrayelement proteins[40];
/*find hash code*/
int findhashCode(char firstLetter, char lastLetter)
{
	return ((int(firstLetter) - 65) + (2 * (int(lastLetter) - 65))) % 40;
}
/* check if given protein exists in structure*/
bool is_Found(char pro[40], int nelem)
{
	for (int k = 0; k < nelem; k++)
	{
		if (strcmp(pro, proteins[k].protein) == 0)
		{
			proteins[k].count = proteins[k].count + 1;
			return true;
		}
	}
	return false;
}
/*main method*/
int main()
{
	/*file open*/
	ifstream fIn("C:/Users/roman/OneDrive/Desktop/proteins.txt");
	char pro[30];
	int n;
	int k;
	int nelem = 0;
	int found = 0;
	/*initializes the protein structure*/
	for (k = 0; k < 40; k++)
	{
		strcpy_s(proteins[k].protein, " ");
		proteins[k].count = 0;
	}
	/*read until end of file*/
	while (!fIn.eof())
	{
		found = 0;
		fIn >> pro;
		/* enter element if not found in the structure*/
		if (!is_Found(pro, nelem))
		{
			n = strlen(pro);
			int k = findhashCode(pro[0], pro[n - 1]);
```

```
                        nelem++;
                        while (1)
                        {
                              if (k >= 40)
                                    k = 0;
                              if (strcmp(proteins[k].protein, " ") == 0)
                              {
                                    strcpy_s(proteins[k].protein, pro);
                                    proteins[k].count = proteins[k].count + 1;
                                    break;
                              }
                              k++;
                        }


                  }

            }
            cout << endl;
            /*print the structure*/
            for (int k = 0; k < 40; k++)
            {
                  if (strcmp(proteins[k].protein, " ") != 0)

                        cout << proteins[k].protein << " " << proteins[k].count << endl;
            }
            system("pause");
            return 0;
}
```

Screenshot of output



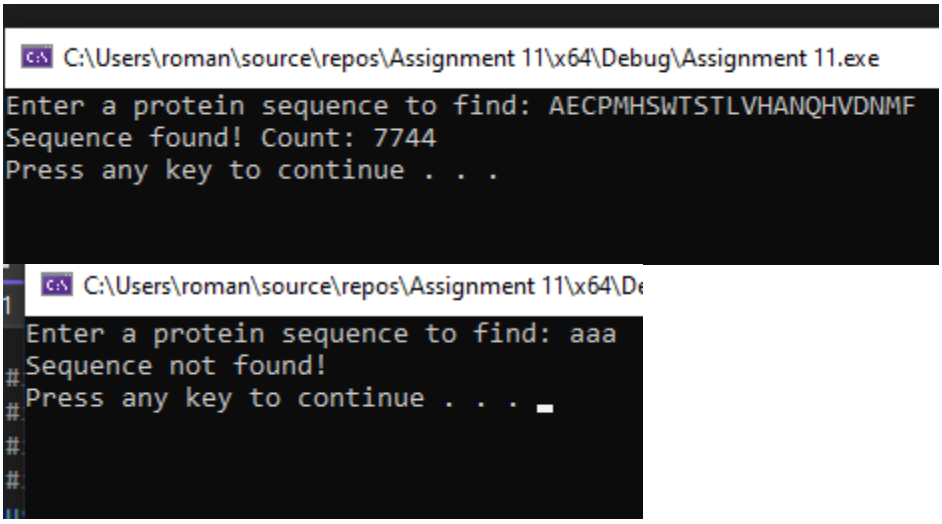
//Code with user input

```
#include<iostream>
#include<string.h>
#include<cstdlib>
#include<fstream>
using namespace std;

/* structure to hold protein sequence and its count */
struct arrayelement
{
    char protein[30];
    int count;
};
```

```
arrayelement proteins[40];

/* find hash code */
int findhashCode(char firstLetter, char lastLetter)
{
    return ((int(firstLetter) - 65) + (2 * (int(lastLetter) - 65))) % 40;
}

/* check if given protein exists in structure */
bool is_Found(char pro[40], int nelem)
{
    for (int k = 0; k < nelem; k++)
    {
        if (strcmp(pro, proteins[k].protein) == 0)
        {
            proteins[k].count = proteins[k].count + 1;
            return true;
        }
    }
    return false;
}

/* main method */
int main()
{
    ifstream fIn("C:/Users/roman/OneDrive/Desktop/proteins.txt");
    char pro[30];
    int n;
    int k;
    int nelem = 0;
    int found = 0;

    for (k = 0; k < 40; k++)
    {
        strcpy_s(proteins[k].protein, " ");
        proteins[k].count = 0;
    }

    while (!fIn.eof())
    {
        found = 0;
        fIn >> pro;

        if (!is_Found(pro, nelem))
        {
            n = strlen(pro);
            int k = findhashCode(pro[0], pro[n - 1]);
            nelem++;
            while (1)
            {
                if (k >= 40)
                    k = 0;
                if (strcmp(proteins[k].protein, " ") == 0)
                {
                    strcpy_s(proteins[k].protein, pro);
                    proteins[k].count = proteins[k].count + 1;
                    break;
                }
```

```
                k++;
            }
        }
    }

    // User input and searching
    char user_input[30];
    cout << "Enter a protein sequence to find: ";
    cin >> user_input;

    // Search for user input sequence
    bool sequence_found = false;
    for (int i = 0; i < 40; i++)
    {
        if (strcmp(proteins[i].protein, user_input) == 0)
        {
            cout << "Sequence found! Count: " << proteins[i].count << endl;
            sequence_found = true;
            break;
        }
    }

    if (!sequence_found)
        cout << "Sequence not found!" << endl;

    // Keep the program open until user input
    system("pause");

    return 0;
}
```

Screenshot of output

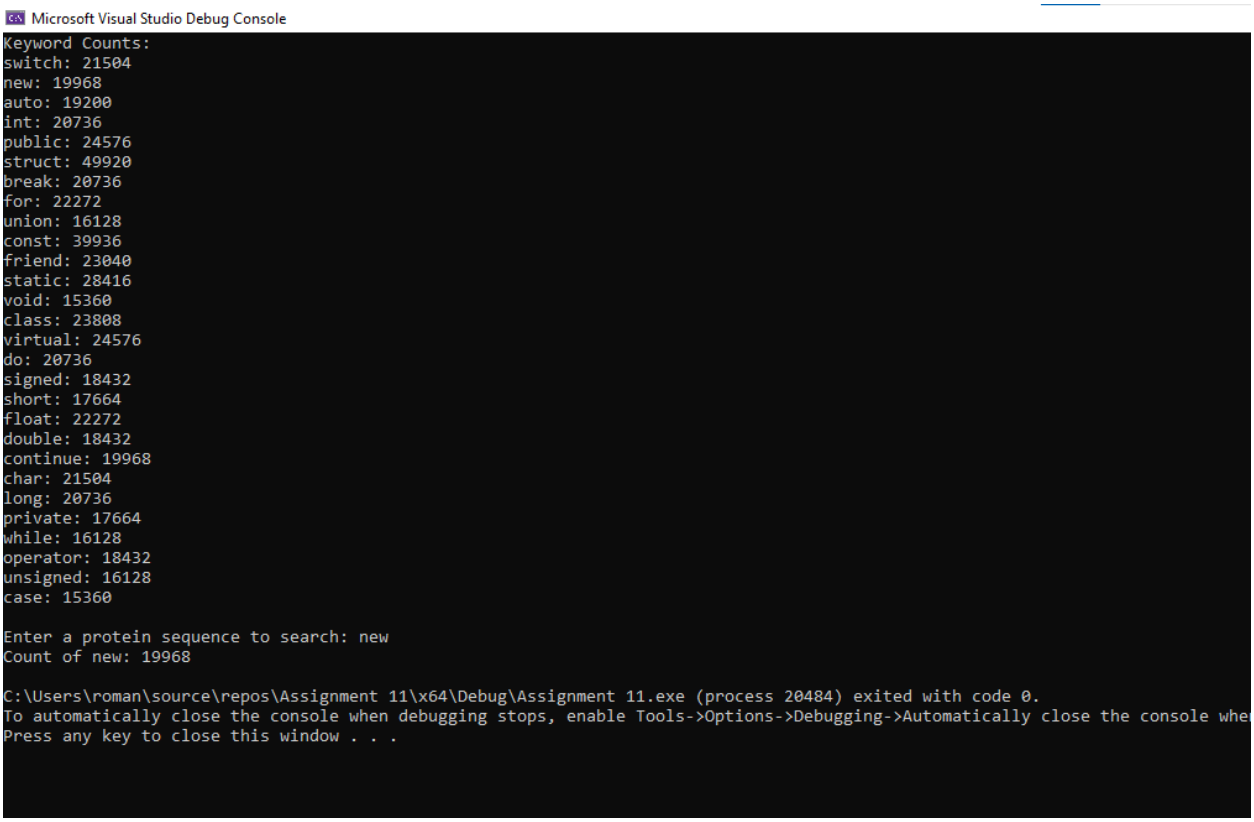


2nd part

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <unordered_map>

int hashFunction(const std::string& keyword) {
    int sum = 0;
    for (char ch : keyword) {
        sum += ch - 'a';
    }
    return sum % 26; // Assuming array size is 26
}

void createPerfectHashTable(const std::vector<std::string>& keywords,
std::vector<std::string>& hashTable) {
    std::unordered_map<int, std::vector<std::string>> buckets; // Buckets for
collision resolution

    for (const std::string& keyword : keywords) {
        int hash = hashFunction(keyword);
        buckets[hash].push_back(keyword);
    }

    // Resolve collisions in buckets
    for (auto& bucket : buckets) {
        if (bucket.second.size() > 1) {
            int offset = keywords.size();
            while (true) {
                bool conflict = false;
                for (const std::string& keyword : bucket.second) {
                    int hash = (hashFunction(keyword) + offset) % keywords.size();
                    if (hashTable[hash] != "") {
                        conflict = true;
                        break;
                    }
                }
                if (!conflict) {
                    for (const std::string& keyword : bucket.second) {
                        int hash = (hashFunction(keyword) + offset) %
keywords.size();
                        hashTable[hash] = keyword;
                    }
                    break;
                }
                ++offset;
            }
        }
        else {
            int hash = hashFunction(bucket.second[0]);
            hashTable[hash] = bucket.second[0];
        }
    }
}
```

```
std::string retrieveKeyword(const std::vector<std::string>& hashTable, const
std::string& keyword) {
    int hash = hashFunction(keyword);
    if (hash >= 0 && hash < hashTable.size()) {
        if (hashTable[hash] == keyword) {
            return hashTable[hash];
        }
        else {
            return "Keyword not found";
        }
    }
    return "Invalid hash value";
}

int main() {
    std::ifstream inputFile("C:/Users/roman/OneDrive/Desktop/keywords.txt");
    if (!inputFile) {
        std::cerr << "Error: Unable to open input file\n";
        return 1;
    }

    std::vector<std::string> keywords;
    std::string word;
    while (inputFile >> word) {
        keywords.push_back(word);
    }
    inputFile.close();

    std::vector<std::string> hashTable(keywords.size());
    createPerfectHashTable(keywords, hashTable);

    // Count occurrences of each keyword
    std::unordered_map<std::string, int> keywordCounts;
    for (const std::string& keyword : keywords) {
        ++keywordCounts[keyword];
    }

    // Print the count of each keyword
    std::cout << "Keyword Counts:\n";
    for (const auto& pair : keywordCounts) {
        std::cout << pair.first << ": " << pair.second << std::endl;
    }

    // Prompt user to input a protein sequence
    std::string inputSequence;
    std::cout << "\nEnter a protein sequence to search: ";
    std::cin >> inputSequence;

    // Search for the input sequence in the hash table
    std::string result = retrieveKeyword(hashTable, inputSequence);
    if (result != "Keyword not found") {
        std::cout << "Count of " << inputSequence << ": " << keywordCounts[result]
<< std::endl;
    }
    else {
        std::cout << "Sequence not found in the hash table.\n";
    }
```

```
    return 0;
}
```


Microsoft Visual Studio Debug Console
Keyword Counts:
switch: 21504
new: 19968
auto: 19200
int: 20736
public: 24576
struct: 49920
break: 20736
for: 22272
union: 16128
const: 39936
friend: 23040
static: 28416
void: 15360
class: 23808
virtual: 24576
do: 20736
signed: 18432
short: 17664
float: 22272
double: 18432
continue: 19968
char: 21504
long: 20736
private: 17664
while: 16128
operator: 18432
unsigned: 16128
case: 15360
Enter a protein sequence to search: new
Count of new: 19968
C:\Users\roman\source\repos\Assignment 11\x64\Debug\Assignment 11.exe (process 20484) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console whe
Press any key to close this window . . .