

1.Task topic

Assignment 9 Anagrams. The user enters a string. Write a program that checks a (given) dictionary for occurrences of anagrams of this string. For example, ANGLE and GALEN are anagrams.

2.Project analysis

Project was based on creating few steps of coding. First, make it possible to obtain string from the user – it will be a word for which anagrams will be taken from the dictionary (File with english words). Then it was necessary to write function which task was checking if pointed word is anagram for user's word or not. To find it out, we know that words are anagrams if same letters occur same number of times. Algorithm tries to find how many times characters appear in the strings and then comparing their corresponding counts. When I coded these two important parts of program, it was time for creating main loop which will use mentioned function for whole list of words and write out anagrams.

3.External specification

To run the program there is required file dictionary.txt, there is stored list of english words, about 40 thousands of words. It is necessary to run program correctly, if there wouldn't be this file, whole project couldn't compile. It happens because in this task, we compare user's word with already known words. There are no needed additionally files. Also specification of input is simple. After running compiler users is asked to provide string without spaces or enters.

4.Internal specification

```
int check_anagram(char a[], char b[]) -function for checking whether strings are anagrams
int first[26] = {0}, second[26] = {0}, c = 0; -filling tables with 0, since number of repeated letters
is 0. In mentioned function next while loops are for checking occurrence of letters.
int word_max_lenght =30; - maximum length of user's string input
char *a1 = (char *)malloc(word_max_lenght*sizeof(char)); - allocating memory for user string
char *a2 = (char *)malloc(word_max_lenght*sizeof(char)); allocating memory for string which was
taken from the dictionary
int *asciitable_dc = (int *)malloc(word_max_lenght*sizeof(int));
it was int table to store data from dictionary (easier to operate on letters as numbers in ASCII)
while ((check = fgetc(fp)) != EOF) till pointer didnt reach end of file
if (check == '\n')- since words were seperated by enters we had to get one by one(1 word in 1 line)
fgetc(a2, word_max_lenght, fp); command which 'takes' word_max_lenght(number) of characters(by
pointer) and put it into previously created table.
asciitable_dc[k] = a2[k] changing from char to int
```

```
if(asciitable_dc[0]>64&&asciitable_dc[0]<91)
    asciitable_dc[0]=asciitable_dc[0]+32;
a2[0]=asciitable_dc[0];
```

Upper lines of code was used because in dictionary there are also words which start with capital letter so first, I had to convert first letter (sometimes capital) into small one.

if (check_anagram(a1, a2)) if everything was fine, it was time for calling function which was checking if word in table from the dictionary is anagram for user's one.

5.Source Code

/* Assignment 9

Anagrams. The user enters a string. Write a program that checks a (given) dictionary for occurrences of anagrams of this string. For example, ANGLE and GALEN are anagrams.

*/

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <ctype.h>
```

```
#include <string>
```

```
#pragma warning(disable:4996)
```

```
int check_anagram(char a[], char b[])
```

```
{  
    int first[26] = {0}, second[26] = {0}, c = 0;
```

```
    while (a[c] != '\0')
```

```
    {  
        first[a[c]-'a']++;  
        c++;  
    }
```

```
    c = 0;
```

```
    while (b[c] != '\0')
```

```
    {  
        second[b[c]-'a']++;  
        c++;  
    }
```

```
    for (c = 0; c < 26; c++)
```

```
    {  
        if (first[c] != second[c])  
            return 0;  
    }
```

```
    return 1;
```

```
}
```

```
int main()
```

```
{
```

```
    int word_max_lenght = 30;
```

```
    int conv;
```

```
    char *a1 = (char *)malloc(word_max_lenght*sizeof(char));
```

```
    char *a2 = (char *)malloc(word_max_lenght*sizeof(char));
```

```
    int *asciitable_dc = (int *)malloc(word_max_lenght*sizeof(int));
```

```
    printf("Enter string\n");
```

```
    scanf("%s", a1);
```

```
    int k=0;
```

```

printf("These are anagrams:\n");

FILE * fp = fopen("dictionary.txt", "r");
if (fp == NULL)
{
    printf("Something is wrong with file. Please correct data.");
    return -1;
}
else
{
    int check;
    int h=0;
    while ((check = fgetc(fp)) != EOF)
    {
        if (check == '\n')
        {
            fgets(a2, word_max_lenght, fp);
            while (a2[k] != '\0') //till wordlist[k] is not equal end
            {
                asciitable_dc[k] = a2[k]; // filling asciitable with data from wordlist
                k++;
            }
            k=0;
            if(asciitable_dc[0]>64&&asciitable_dc[0]<91)
                asciitable_dc[0]=asciitable_dc[0]+32;
            a2[0]=asciitable_dc[0];
            if ( check_anagram(a1, a2))
            {
                printf("%s\n", a2);
                h++;
            }

        }

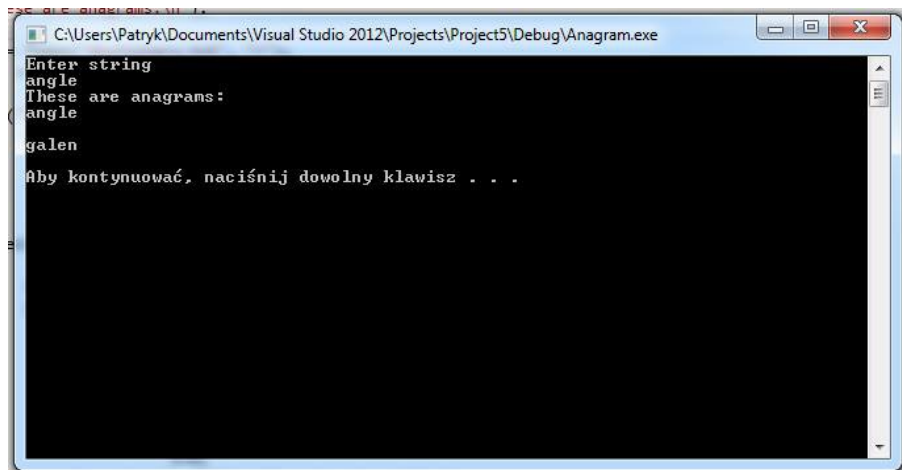
    }

    if (h < 1)
    {
        printf("\nNo anagrams can be found\n");
    }
    system ("pause");
    return 0;
}
}

```

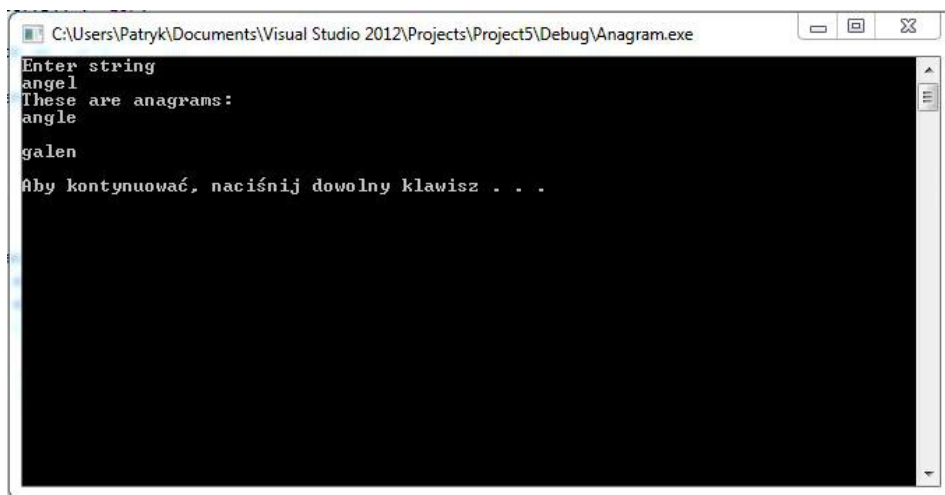
6. Testing

After coding important parts mentioned in project analysis I wanted to test if it works correctly. In task there was word angle to check if there exists anagram and this is a result. In output there is same word found in list(anagram for word can be this word itself)



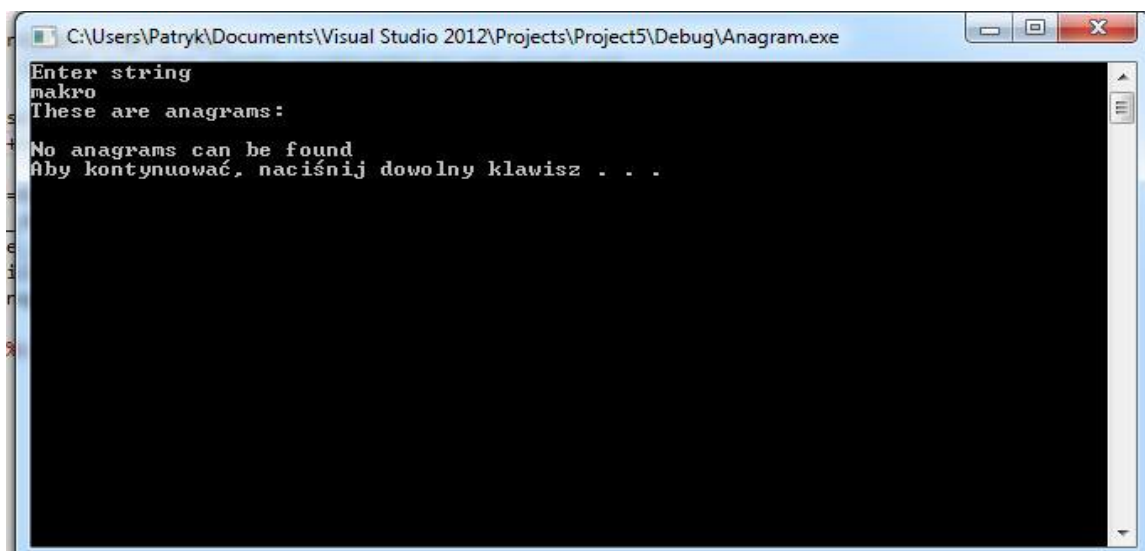
```
C:\Users\Patryk\Documents\Visual Studio 2012\Projects\Project5\Debug\Anagram.exe
Enter string
angle
These are anagrams:
angle
galen
Aby kontynuować, naciśnij dowolny klawisz . . .
```

In dictionary there wasn't word angel (that's why it didnt appear). But when we wrote angel, anagrams were angle and galen.



```
C:\Users\Patryk\Documents\Visual Studio 2012\Projects\Project5\Debug\Anagram.exe
Enter string
angel
These are anagrams:
angle
galen
Aby kontynuować, naciśnij dowolny klawisz . . .
```

And here is exemplary output when nothing is found.



```
C:\Users\Patryk\Documents\Visual Studio 2012\Projects\Project5\Debug\Anagram.exe
Enter string
makro
These are anagrams:
No anagrams can be found
Aby kontynuować, naciśnij dowolny klawisz . . .
```

7. Conclusions

The whole programm in my opinion is ready. Problems may appear

when user's input is not correctly. Code can be improved by creating safety rules of input-additional conditions for string input.