Лабораторна робота 1
З дисципліни “Системне програмування”
Студента групи МІ-31
Бідзілі Святослав Олексійовича

Постановка задачі:

Нехай маємо текстовий файл, на який не накладаються обмеження щодо його розміру. Ми також не накладаємо обмеження на довжину рядка в цьому файлі.
Текст складається зі слів, наприклад, ідентифікатори англійської мови. Слова перемежовуються проміжками, дужками, кодами операцій, взагалі символами, що природно відділяють слова одне від одного. Ми також не будемо займатися питанням правопису таких слів. Нехай на довжину слова ми встановимо обмеження – 30 літер.
В поле результату потрібно вивести слова *без повторень*, що задовольняють певній умові.
Щодо файла, вимоги до якого сформульовані вище, виконати наступні лабораторні роботи (варіант – один з 13, обрати згідно порядкового номеру у списку групи; 14 варіант узгоджується з викладачем попередньо для додаткових балів):

1. Знайти всі слова максимальної довжини.

Виконання

Весь код для лабораторної роботи розміщений за посиланням:

https://github.com/anakib1/KNU_SP_2023_lab1

Розглянемо код за функціями:

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

    FILE *inputFile = fopen("input.txt", "r");
    FILE *outputFile = fopen("output.txt", "w");
    if (inputFile == NULL) {
        perror("Error opening input.txt");
        return 1;
    }
```

Ми відкриваємо вхідний/вихідний файл, перевіряємо їх наявність.

```
wchar_t longestWord[mxlen] = L"";
wchar_t currentWord[mxlen] = L"";

wchar_t longestWords[mxlen][mxlen];
int numLongestWords = 0;

while (fwscanf(inputFile, L"%99ls", currentWord) != EOF) {
    int currentLength = wcslen(currentWord);
    int longestLength = wcslen(longestWord);

    long long hash = calculateCurrentHash(currentWord);

    if (currentLength > longestLength) {
        numLongestWords = 0;
        wcscpy(longestWord, currentWord);
        add(hash);
        wcscpy(longestWords[numLongestWords++], currentWord);
    } else if (currentLength == longestLength) {
        if (!was(hash)) {
            wcscpy(longestWords[numLongestWords++], currentWord);
            add(hash);
        }
    }
}

fclose(inputFile);
```

В масиві longestWord, ми зберігаємо найдовше слово (яке ми поки що бачили). В масиві currentWord ми завжди записуємо слово яке зараз зчитуємо з вхідного файлу. В масиві longestWords ми зберігаємо результат – всі слова найдовшої довжини. Також ми підтримуємо кількість слів в ньому, онуляючи її при виявлені нового найдовшого слова.

Всі введення виведення виконуються через wchar для того, щоб правильно працювати з кирилицею в вхідних файлах.

Також додатково, для збереження унікальності слів у відповіді будемо користуватися хеш-функцією – додавати слово в відповідь, лише якщо його хеш ми ще не бачили. Наївна реалізація виглядає ось так:

```
// sanity bound 1e6
#define mxn 1000000L
#define M 70000L

long long hashMap[M][2 * (int)(mxn/M)];
int lens[M];

bool was(long long hash) {
    int id = hash % M;
    for (int i = 0; i < lens[id]; i ++) {
        if (hashMap[id][i] == hash) {
            return true;
        }
    }
    return false;
}

void add(long long hash) {
    int id = hash % M;
    hashMap[id][lens[id]] = hash;
    lens[id]++;
}
```

```
#define mxlen 30

#define BASE 117L
#define MOD 1000000009L

long long calculateCurrentHash(wchar_t *word) {
    long long ret = 0;
    for(int i = 0; i < wcslen(word); i ++ ) {
        ret = (ret * BASE + word[i]) % MOD;
    }
    return ret;
}
```

Ми використовуємо звичайний поліноміальний хеш і двовимірну хеш-таблицю, яка дає перевірку за O(mxn/N) в середньому, що має відносно швидку для нашої задачі асимтотику.

Після проходження по всьому файлу, ми виводимо відповідь в вихідний файл, знову ж таки за допомогою широких чарів.

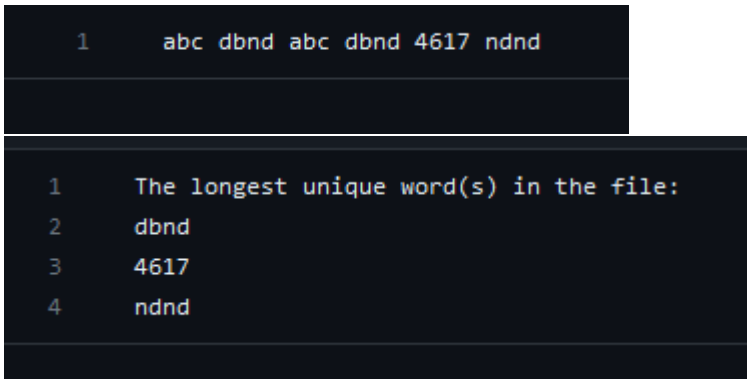
```c
if (wcslen(longestWord) > 0) {
    fprintf(outputFile, "The longest unique word(s) in the file:\n");
    for (int i = 0; i < numLongestWords; i++) {
        fwprintf(outputFile, L"%ls\n", longestWords[i]);
    }
} else {
    fprintf(outputFile, "Input is empty.\n");
}

fclose(outputFile);
```

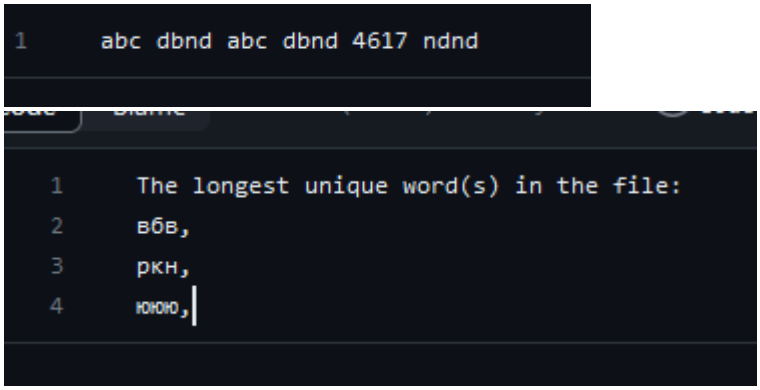
Сумарно, ми лише один раз проходимося по вхідному файлу, копіюємо кожне слово не більше одного разу, тому маємо лінійну складність коду від розміру вхідного файлу. Варто зазначити, що ми не зберігаємо в пам’яті всі слова файлу, а підтримуємо множину лише найдовших слів.

Приклади роботи:

Приклад 1



Приклад 2



Висновок:

Під час виконання роботи використавши мову С я зміг реалізувати просту обробку текстових файлів з розбиттям на рядки.