Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

Про виконання лабораторної роботи № 7

з навчальної дисципліни

“Базові методології та технології програмування”

ПРОГРАМНА РЕАЛІЗАЦІЯ ОБРОБЛЕННЯ МАСИВІВ ДАНИХ ТА СИМВОЛЬНОЇ ІНФОРМАЦІЇ

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

ВИКОНАВ

студент академічної групи КБ-20

Василенко К. О.

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Поліщук Л. І.

Кропивницький – 2021

**Мета роботи** полягає у набутті ґрунтовних вмінь і практичних навичок синтезу алгоритмів оброблення масивів даних та символьної (текстової) інформації у кодуваннях UTF-8 і CP866, їх програмної реалізації мовою програмування мовою програмування С (ISO/IEC 9899:2018) задля реалізації програмних засобів у вільному кросплатформовому Code::Blocks IDE.

**Завдання до лабораторної роботи:**

1. Реалізувати програмний забезпечення розв’язування задачі 7.1.
2. Реалізувати програмне забезпечення розв’язування задачі 7.2.

**Варіант № 16**

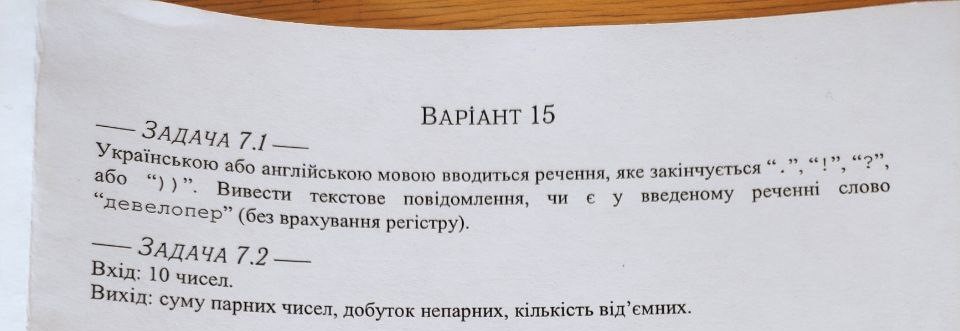


Рисунок 1 – Варіант № 15

**Завдання 1**

* 1. **Формалізація задачі**
     1. **Строга постановка задачі**

**Вихідні данні:**

Chek\_developer – перевіряє наявність слова в рядку

Word\_transform – перевіряє наявності “.”, ”!” в реченні.

**Вхідні данні:**

array – строка яку вводить користувач у консолі.

* 1. **Лістинг до завдання 1**

#include <stdlib.h>

#include <string.h>

int main()

{

system("chcp 1251 &cls");

char array[1000];

int Chek\_developer = 0;

printf("Введіть рядок символів: ");

gets(array);

for (int i = 0; i < strlen(array); i++)

{

if (array[i] == 'девелопер')

Chek\_developer ++;

}

for (int i = 0; i < strlen(array); i++)

{

if (array[i] == '))', '.','!','?')

break;

}

if (Chek\_developer == 0){

printf("В цьому реченні немає слова “девелопер”");}

else{ printf("В цьому реченні наявне слово “девелопер”= %d", Chek\_developer);}

return 0;

}

Тестування цього лістингу показано у «Додатку 1»

**Завдання 2**

* 1. **Формалізація задачі**
     1. **Строга постановка задачі**

**Вихідні данні:**

hh – сума непарних чисел;

gg – кількість від’ємніх чисел;

ss – добуток парних чисел масиву.

**Вхідні данні:** hs – масив випадкових цілих чисел.

* 1. **Лістинг до завдання 2**

#include <stdio.h>

#include <stdlib.h>

#include<locale.h>

int main()

{

setlocale(LC\_ALL, "ukr");

printf("Программа призначена для упорядкування елементiв масиву.\

\nВиконав студент КБ-20 Василенко Костянтин Олегович.\

\nEmail: kostyavasilenko9@gmail.com || Phone number: +380978529358\n\n");

{

int hh;

int ss;

int gg;

int hs[] = { 5,7,6,3,9,12,4,53,11,1 };

for (int hs = 0; hs < 10;hs++) {

if (hs % 2 == 1) {

hh ==hs+hs;

}

if (hs % 2 == 0) {

ss== hs\*hs;

}

if (hs==-hs) {

gg + 1;

}

}

cout << hh;

cout << ss;

cout << gg;

}

Тестування цього лістингу показано у «Додатку 2»

**Додаток 1**

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | BMTP-LAB7-Vasilenko |
| Назва проекта / ПЗ  Name of Project / Software | Vasilenko-task\_7\_1.exe |
| Рівень тестування  Level of Testing | системний  /  System Testing |
| Автор тест-сьюта  Test Suite Author | Василенко Костянтин Олегович |
| Виконавець  Implementer | Василенко Костянтин Олегович |

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест-кейса / Test Case ID | Дії (кроки) /  Action  (Test Steps) | Очікуваний  результат /  Expected Result | Результат тестування (пройшов/не вдалося/ заблокований) /  Test Result (passed/failed/ blocked) |
| TC-01 | 1. Відкрити застосунок | Программа призначена для пошуку слова "девелопер" у речені.  Виконав студент КБ-20 Василенко Костянтин Олегович. Email: kostyavasilenko9@gmail.com || Phone number: +380978529358 Введіть рядок: | passed |
| TC-02 | 1. Відкрити застосунок 2. Ввести “девелопер.” | В цьому реченні наявне слово “девелопер” | passed |
| TC-03 | 1. Відкрити застосунок 2. Ввести “що таке девелопер? ” | В цьому реченні наявне слово “девелопер” | passed |
| TC-04 | 1. Відкрити застосунок 2. Ввести “чому записано саме це речення? ” | В цьому реченні немає слова “девелопер” | passed |
| TC-05 | 1. Відкрити застосунок 2. Ввести “девелопер серед нас))” | В цьому реченні наявне слово “девелопер” | passed |
| TC-06 | 1. Відкрити застосунок 2. Ввести “девелоперрр” | В цьому реченні немає слова “девелопер” | passed |
| TC-07 | 1. Відкрити застосунок 2. Ввести “I love developer))” | В цьому реченні немає слова “девелопер” | passed |
| ТС-08 | 1. Відкрити застосунок 2. Ввести “хто цей dевелопер?” | В цьому реченні немає слова “девелопер” | passed |
| ТС-09 | 1. Відкрити застосунок 2. Ввести “девелопер” | В цьому реченні немає слова “девелопер” | passed |
| ТС-10 | 1. Відкрити застосунок 2. Ввести “український девелопер!” | В цьому реченні наявне слово “девелопер” | passed |

**jДодаток 2**

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | BMTP-LAB7-Vasilenko |
| Назва проекта / ПЗ  Name of Project / Software | Vasilenko-task\_7\_2.exe |
| Рівень тестування  Level of Testing | системний  /  System Testing |
| Автор тест-сьюта  Test Suite Author | Василенко Костянтин Олегович |
| Виконавець  Implementer | Василенко Костянтин Олегович |

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест-кейса / Test Case ID | Дії (кроки) /  Action  (Test Steps) | Очікуваний  результат /  Expected Result | Результат тестування (пройшов/не вдалося/ заблокований) /  Test Result (passed/failed/ blocked) |
| TC-01 | 1. Відкрити застосунок | Программа призначена для пошуку слова "девелопер" у речені.  Виконав студент КБ-20 Василенко Костянтин Олегович. Email: kostyavasilenko9@gmail.com || Phone number: +380978529358 Введіть рядок: | passed |
| TC-02 | 1. Відкрити застосунок | Сума парних чисел, добуток непарних, кількість від’ємніх. | passed |
| TC-03 | 1. Відкрити застосунок | Сума парних чисел, добуток непарних, кількість від’ємніх. | passed |
| TC-04 | 1. Відкрити застосунок | Сума парних чисел, добуток непарних, кількість від’ємніх. | passed |
| TC-05 | 1. Відкрити застосунок | Сума парних чисел, добуток непарних, кількість від’ємніх. | passed |
| TC-06 | 1. Відкрити застосунок | Сума парних чисел, добуток непарних, кількість від’ємніх. | passed |
| TC-07 | 1. Відкрити застосунок | Сума парних чисел, добуток непарних, кількість від’ємніх. | passed |
| ТС-08 | 1. Відкрити застосунок | Сума парних чисел, добуток непарних, кількість від’ємніх. | passed |
| ТС-09 | 1. Відкрити застосунок | Сума парних чисел, добуток непарних, кількість від’ємніх. | passed |
| ТС-10 | 1. Відкрити застосунок | Сума парних чисел, добуток непарних, кількість від’ємніх. | passed |

**Висновок**

Під час виконання лабораторної роботи №7 мною було набуто ґрунтових вмінь та практичних навичок синтезу алгоритмів оброблення інформації, даної в масивах даних та символьної інформації, зокрема у вигляді тексту, у кодуваннях UTF-8 та CP866, їх програмної реалізації мовою програмування С (ISO/IEC 9899:2018) для реалізації програмних засобів у вільному кросплатформовому Code::Blocks IDE.

На стадії написання коду основною проблемою стало те, що OS Windows не вміє «адекватно» працювати із багатобайтними кодуваннями. Ця операційна система може працювати «виключно» у кодуванні UTF-16. Працювати у UTF-8 неможливо в основному тому, що Windows не допускає UTF-8 в якості системної кодової сторінки ANSI, навіть якщо вона має кодову сторінку ANSI для UTF-8, кодову сторінку 65001. Здається, є кілька причин для цього:

- У Windows є одна кодова сторінка ANSI для кожного підтримуваного мови, на відміну від Unix подібних систем, де мова і кодування можуть бути встановлені незалежно.

- Коли Unicode був новим, Microsoft вирішила, що UCS-2 буде кращим способом підтримки Unicode. У той час Unicode був 16-бітовим.

-Кодова сторінка 65001 (UTF-8 це CP 65001) не працює всюди. Зокрема, він не працює з деякими функціями для оброблення багатобайтних символів у Windows, де багатобайтові символи вимагають один або два байти, тоді як UTF-8 вимагає від одного до чотирьох байтів.

Ось що говорять розробники Windows з цього поводу: « Оскільки Windows працює в UTF-16 ( WCHAR), вам може знадобитися перетворити дані UTF-8 у UTF-16 (або навпаки), щоб взаємодіяти з API Windows.»

Для переведення введених данних з UTF-16 в UTF-8 Microsoft рекомендує використовувати такі дві функції реалізовані у бібліотеці windows.h: [MultiByteToWideChar](https://docs.microsoft.com/en-us/windows/desktop/api/stringapiset/nf-stringapiset-multibytetowidechar) та [WideCharToMultiByte](https://docs.microsoft.com/en-us/windows/desktop/api/stringapiset/nf-stringapiset-widechartomultibyte).

[MultiByteToWideChar](https://docs.microsoft.com/en-us/windows/desktop/api/stringapiset/nf-stringapiset-multibytetowidechar) і [WideCharToMultiByte](https://docs.microsoft.com/en-us/windows/desktop/api/stringapiset/nf-stringapiset-widechartomultibyte) дозволяє конвертувати між UTF-8 і UTF-16 ( WCHAR) (і інших кодових сторінок). Це особливо корисно, коли застарілий API Win32 може розуміти лише WCHAR. Ці функції дозволяють перетворити вхід UTF-8 WCHAR на перехід в API -W, а потім перетворити будь-які результати назад, якщо це необхідно. При використанні цих функцій із CodePageвстановленим значенням CP\_UTF8 використовується dwFlags або 0 , або MB\_ERR\_INVALID\_CHARS, інакше  ERROR\_INVALID\_FLAGS.

Але під час реалізації програмного застосунку до задачі 7.1 з’явилася така проблема: функції для «читання» розширених символів з потоку вводу нормально «зчитували» лише «однобайтні» символи ( наприклад функція wscanf(), \_getws() реалізовані у бібліотеці wchar.h ). Багатобайтні символи просто інтерпретувались як нуль символи («\0»).

Щоб вирішити дану проблему було прийнято рішення застосувати функцію реалізовану у бібліотеці «windows.h» ReadConsole().

**Реалізація функції ReadConsole():**

BOOL WINAPI ReadConsole(

\_In\_     HANDLE  hConsoleInput,

\_Out\_    LPVOID  lpBuffer,

\_In\_     DWORD   nNumberOfCharsToRead,

\_Out\_    LPDWORD lpNumberOfCharsRead,

\_In\_opt\_ LPVOID  pInputControl

);

**Параметри функції:**

* hConsoleInput [in]  
  Дескриптор вхідного буфера консолі. Дескриптор повинен мати право доступу **GENERIC\_READ** . Для отримання додаткової інформації див. [Захист консольного буфера та права доступу](https://docs.microsoft.com/en-us/windows/console/console-buffer-security-and-access-rights) .
* lpBuffer [out]  
  Вказівник на буфер, який приймає дані, прочитані з вхідного буфера консолі.
* nNumberOfCharsToRead [in]  
  Кількість символів, які потрібно прочитати. Розмір буфера, на який вказує параметр lpBuffer, повинен бути не менше nNumberOfCharsToRead \* sizeof(TCHAR)байтів.
* lpNumberOfCharsRead [out]  
  Вказівник на змінну, яка отримує кількість фактично прочитаних символів.
* pInputControl [in, optional]  
  Вказівник на структуру [**CONSOLE\_READCONSOLE\_CONTROL,**](https://docs.microsoft.com/en-us/windows/console/console-readconsole-control) яка визначає керуючий символ для сигналізації закінчення операції зчитування. Цей параметр може мати значення **NULL** .

Цей параметр за замовчуванням вимагає введення Unicode. Для режиму ANSI встановіть для цього параметра значення **NULL** .

Повертаєме значення

-Якщо функція успішна, повертається значення відмінне від нуля.

-Якщо функція виходить з ладу, повернене значення дорівнює нулю.

ReadConsole читає введення з клавіатури з буфера введення консолі. Він поводиться як функція ReadFile , за винятком того, що він може читати як в Unicode (широкосимвольний), так і в режимі ANSI. Щоб мати програми, які підтримують єдиний набір джерел, сумісних з обома режимами, використовують ReadConsole, а не ReadFile . Хоча ReadConsole можна використовувати лише з дескриптором вхідного буфера консолі, ReadFile можна використовувати з іншими дескрипторами (наприклад, файлами або конвеєрами ). ReadConsole не працює, якщо використовується зі стандартним дескриптором, який був перенаправлений як щось інше, ніж консольний дескриптор.