**Звіт по лабораторній роботі №7**

**Тема**: Функції.

**Розробник**: студентка Зеленець Олена, група КІТ-120а.

**Перевірив**: асистент Челак Віктор Володимирович.

**Загальне завдання**: Реалізувати програми з використанням функції:

1. Визначити, чи є ціле 6-значне число «щасливим» квитком («щасливий квиток» - це квиток, у якому сума першої половини чисел номера дорівнює сумі другої половини).

2. У заданому тексті знайти кількість слів за умови, що між словами може бути будь яка кількість пропусків.

3. Реалізувати функцію, що визначає, скільки серед заданої послідовності чисел таких пар, у котрих перше число менше наступного, використовуючи функцію з варіативною кількістю аргументів.

**Опис програми 1:**

1. Оголошуємо функцію під назвою int calculate.

2. У «тілі фунції» int main вказуємо значення змінної A, яке містить у собі номер квитка і вводимо змінну char tic, sum1 та sum2, які відповідають за розрахунок суми перших та останніх 3 чисел 6-значного числа та змінну result, яка відповідає за визначення результату.

3. Далі вказуємо, що змінній result викликається наше значення функції , у якій вказуються умови, при яких наш квиток є «щасливим» чи навпаки.

4. Далі в тілі функції int calculate(int A) записуємо алгоритм дій, при яких будуть відокремлюватися перші та останні три цифри нашого 6-значного числа.

5. І тепер виконується умова, при якій прирівнюється sum1 та sum2 і в результаті рівності result = 1, тобто true, а коли sum1 не дорівнює sum2, result = 0, тобто false.

6. Отже, наш програмний код має вигляд (рис.1):

```c
int calculate(int A);

int main (){
    int A = 144304;
    char tic;
    int sum1 = 0;
    int sum2 = 0;
    int result = 0;
    result = calculate(A);
    if(result = 1){
        tic = 'Y';
    }
    else if(result = 0){
        tic = 'N';
    }
    return 0;
}

int calculate(int A){
    int sum1 = 0;
    int sum2 = 0;
    while (A/1000 > 0 ){
        sum1 += A%10;
        A /= 10;
    }
    int N=A;
    while(N > 0 ){
    sum2 += N%10;
    N /= 10;
    }
    int result;
    if (sum1 == sum2){
        result = 1;
    }
    else if(sum1 != sum2){
        result = 0;
    }
    return result;
}
```

**Рисунок 1 –** Готовий код згідно 1 завдання

7. Ставимо breakpoint на 39 рядку та запускаємо програму через Debug. Проблем не виявлено, правильно знайдено кількість слів у заданому тексті, усе працює. (рис.2)

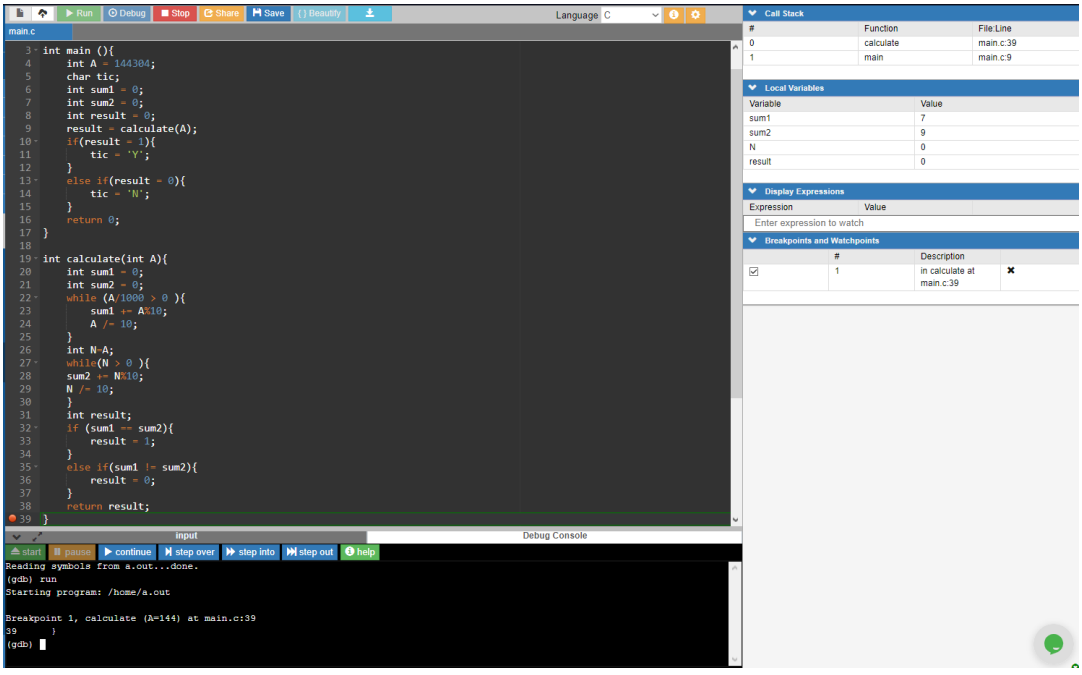


**Рисунок 2** - Результат запуску Debug програми, перевірка правильності коду

**Опис програми 2:**

1. Підключаємо різні бібліотеки для контролю процесу виконання програми, перетворення типів та використання булівських флагів, а також математичних функцій.

2. Через #define задаємо розмір, у нашому випадку 23.

3. Оголошуємо функцію int Count_word(char sentence[]).

4. У тілі int main вводимо змінну char sentence, яка відповідає за розмір і задаємо текст, у якому шукатимемо кількість слів (" One  hundred  ninety  ") та змінній  count_word присвоюємо Count_word(sentence) для того, щоб в змінну count_word записувався результат виконання функції Count_word, в яку переданий аргумент sentence, який зберігає в собі текст для обробки.

5. Далі в тілі функції int Count_word(char sentence[]) вводимо змінну

int count_word = 0;

char placeholder = ' ';

6. І на даному етапі вказуємо, що на bool flag = false.

7. Далі розпочинаємо цикл for, задаємо діапазон відповідно до нашого SIZE і вказуємо умову: якщо елемент нашого тексту не є пробілом і не є закінченням тексту, то flag = true.

8. Також вказуємо, що коли (i + 1) <= SIZE і елемент в тексті не є пробілом, то відбувається рахунок слова.

9. Виконання даного алгоритму дій виконується, доки не переберуться всі елементи тексту.

10. Отже, програмний код має вигляд (рис.3):

```c
#include <stdlib.h>
#include <math.h>
#include <stdbool.h>

#define SIZE 23

int Count_word(char sentence[]);

int main() {
    char sentence[SIZE] = { " One  hundred  ninety  " };
    int count_word = Count_word(sentence);
    return 0;
}
    int Count_word(char sentence[]){
    int count_word = 0;
    char placeholder = ' ';
    bool flag = false;
    for (int i = 0; i < SIZE; i++) {
        if (sentence[i] != placeholder && sentence[i]!= '\0') {
            flag = true;
            while ((i + 1) < SIZE && sentence[i + 1] != placeholder && sentence[i + 1] != '\0')
                i++;
        }
        if (flag) {
            count_word++;
            flag = false;
        }
    return count_word;
    }
 }
```

**Рисунок 3** – Готовий програмний код згідно 2 завдання

11. Ставимо breakpoint на 30 рядку та запускаємо програму через Debug. Проблем не виявлено, правильно знайдено кількість слів у заданому тексті , усе працює. (рис.4)

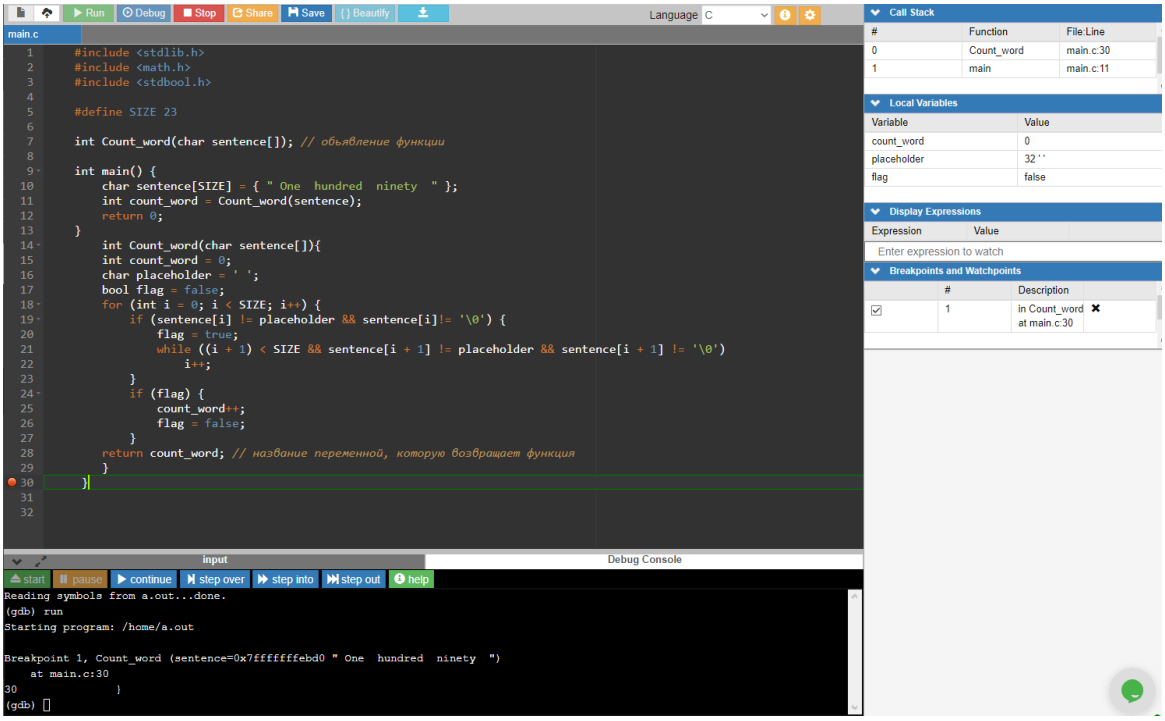


**Рисунок 4** - Результат запуску Debug програми, перевірка правильності коду

**Опис програми 3:**

1. Використовуємо бібліотеку stdarg.h, яка надає можливість реалізації варіативної функції.

2. Оголошуємо функцію int variable(int m, ...).

3. У головному тілі функції int main вказуємо змінну int count, у якій будуть розраховуватися із заданих елементів функції пари чисел, у яких перше число менше наступного. Перший елемент відповідає за кількість елементів функції.

4. Викликаємо функцію int variable(int m, ...) і в тілі цієї функції вказуємо змінні int count = 0; // значення результату

```
int b = 0;
va_list temp;
va_start (temp, m);
```

5. Далі прописуємо умови циклу for, який буде повторюватися, поки i=m, тоді va_arg(temp, int).

6. Тоді вказуємо умова if, у якій порівнюється значення теперішнє та попереднє, а b – це наше попереднє значення.

7. У змінну b записується va_arg(temp, int), тобто число, щоб порівнювати числа в діапазоні далі, поки не виконається умова i=m.

8. Після виконання всіх дій у змінну va_end записується кількість пар, які задовольняють виконання нашої поставленої задачі.

9. Отже, програмний код має вигляд (рис. 5):

```c
#include <stdarg.h>

int variable(int m, ...);

int main(){
    int count = variable(5, 7, 4, 9, -10, 11);
    return 0;
}

int variable(int m, ...){ // виклик фукнкції
    int count = 0; // значення результату
    int b = 0;
    va_list temp;
    va_start (temp, m);
    for (int i = 1; i < m; i++){
        va_arg(temp, int);
        if(va_arg(temp, int) > b){
            count++;
        }
     b = va_arg(temp, int);
    }
    va_end(temp); // записується кількість пар
    return (count);
}
```

**Рисунок 5** – Готовий програмний код згідно 3 завдання

10. Ставимо breakpoint на 24 рядку та запускаємо програму через Debug. Проблем не виявлено, правильно знайдено кількість слів у заданому тексті , усе працює. (рис. 6)

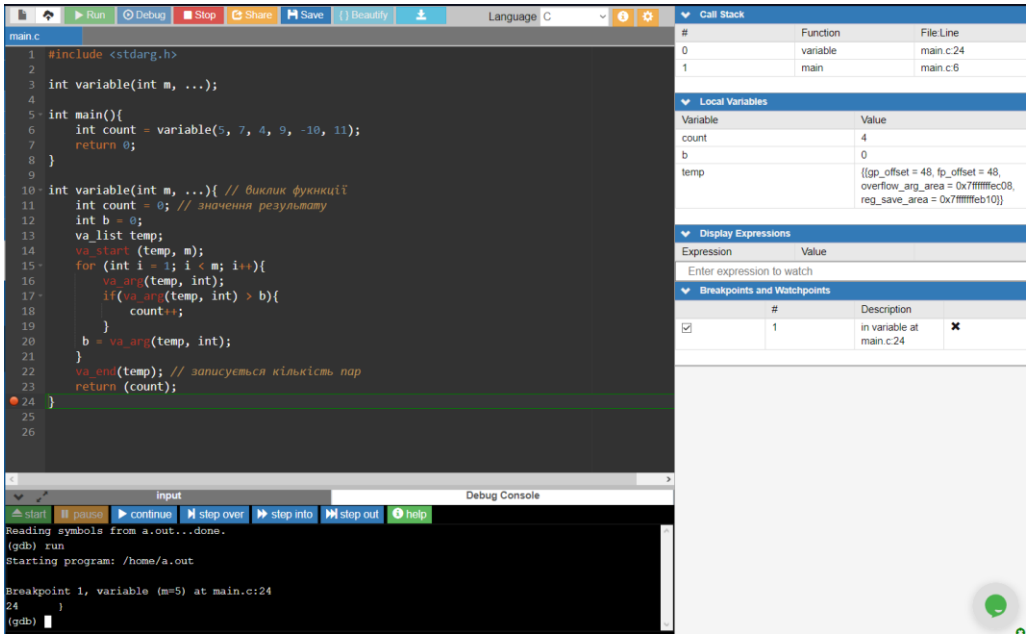


**Рисунок 6** - Результат запуску Debug програми, перевірка правильності коду

Через командну строку Linux додали зміни до майбутнього коміту та запушили всі зміни на GitHub.

**Висновок**:

Для виконання лабораторної роботи ми навчились створювати та реалізовувати алгоритми функції, працювали над тим, щоб переробити код з використанням циклів та масивів на програмний код, який використовуватиме функції для обчислення результату, а також використовували функцію з варіативною кількістю аргументів.