Zajęcia XI

Cele:

zapoznać się z debugowaniem

Problem

Program nie działa jak powinien. Trzeba poprawić, ale żeby to zrobić, trzeba prześledzić działanie, aby znaleźć rozbieżność między obserwacją a założenienm. Można użyć `std::cout` aby wyświetlać zmienne na wyjście, ale można inaczej.

Debugowanie

Większość IDE udostępnia mechanizmy do śledzenia wykonywania programu krok po kroku, podczas którego można obserwować jak zmieniają się zmienne, oraz ustawianie `breakpoint'ów`, miejsc, w których wykonywanie się zatrzyma i umożliwi analizę stanu programu.

Valgrind

Jednym z najczęściej występującym błędem w programach `C/C++` to wyciek pamięci. `Valgrind` umożlwia automatyczną detekcję wycieków pamięci.

Zadanie Lab I

Sprawdzić, jak działa `Valgrind`

Zadanie I

Poniższy program nie działa. Korzystając z debuggera lub/i valgrinda, napraw kod. Opisz (słownie) co zostało zmienione i dlaczego te zmiany naprawiły program. Czy poniższy program zużywa tyle zasobów ile powinien? Uzasadnij.

```c
#include <iostream>
#include <cstring>

/*
 * This function creates a dynamic copy of the input DNA string,
 * removes all occurrences of the substring "ATCG", and computes
 * the GC content on the modified string.
 */
double gc_content_without_atcg(const char *dna)
{
        size_t length = std::strlen(dna);
        char *copy = new char[length + 1];
        std::strcpy(copy, dna);

        size_t write_index = 0;
        for (size_t i = 0; i < length;)
        {
                if (i + 3 < length &&
                        copy[i] == 'A' && copy[i + 1] == 'T' &&
                        copy[i + 2] == 'C' && copy[i + 3] == 'G')
                {
                        i += 4;
                }
                else
                {
                        copy[write_index++] = copy[i++];
                }
        }
        copy[write_index] = '\0';

        size_t gc_count = 0;
        for (size_t i = 0; i < write_index - 1; i++)
        {
                if (copy[i] == 'G' || copy[i] == 'C')
                {
                        gc_count++;
                }
        }

        double gc_content = write_index > 0 ? (double)(gc_count) / write_index : 0.0;
        return gc_content;
}

#define ITERATIONS 100000000

int main()
{
        // This simulate processing large set of DNA sequences
        for (int i = 0; i < ITERATIONS; i++)
        {
                const char *dna = "CATCGGATTC";
                double gc = gc_content_without_atcg(dna);
                std::cout << "GC content without ATCG: " << gc << std::endl;
        }

        return 0;
}
```