# Wstęp do programowania w języku C

Grupa MSz w czwartki

Lista 7 na zajęcia 1.12.2022

---

**Zadanie 1.** ***(10 punktów na pierwszej pracowni, 5 punktów na drugiej)***

Wracamy na chwilę do problemu z listy 5, a dokładniej do jego prostrzej wersji. Oto jedno z ładnych rozwiązań:

```
int n;
long seq[];
unsigned long counter;

void countSumZero(int i, long sum) {
  if (i == 0) {
    if (sum == 0) counter++;
    return;
  }
  i--;
  countSumZero(i, sum);
  countSumZero(i, sum + seq[i]);
}
```

Chcemy je trochę przyspieszyć za pomocą kompilacji dostosowanej pod konkretne dane wejściowe.[1] W przypadku tego algorytmu, znając dane wejściowe możemy zrezygnować z testu `i == 0` a ponadto nie musimy odczytywać liczb z tablicy. Przeanalizuj rozwiązanie wyprodukowane dla pierwszego przykładu:

```
5
1 2 5 -3 -2
void countSumZero0(long sum) { if (sum == 0) counter++; }
void countSumZero1(long sum) { countSumZero0(sum); countSumZero0(sum + -2); }
void countSumZero2(long sum) { countSumZero1(sum); countSumZero1(sum + -3); }
void countSumZero3(long sum) { countSumZero2(sum); countSumZero2(sum + 5); }
void countSumZero4(long sum) { countSumZero3(sum); countSumZero3(sum + 2); }
void countSumZero5(long sum) { countSumZero4(sum); countSumZero4(sum + 1); }
```

[1] Nazywa się to *JIT – Just-In-Time* compilation.

Napisz program, który wczytuje dane do tego problemu ($n$ i sekwencję liczb) i wypisuje na wyjście program w C, który go rozwiąże szybciej.

Na koniec zmierz czas działania zwykłego rozwiązania i generowanego na jakimś większym teście, np.

```
32
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29 30
-195 -140
```

*Uwaga: pamiętaj o włączonej optymalizacji* `-O3` .

**Zadanie 2.** ***(10 punktów)***

Ulepsz rozwiązanie z zadania 1:

1. Kolejne funkcje zdefiniuj przy pomocy makra z parametrami, dzięki czemu kod będzie krótszy.

   *Wskazówka: Przyda się konkatenacja tokenów w makrach.*

2. Przyspiesz jeszcze bardziej: wypróbuj instrukcję `counter+=!sum;` w obu rozwiązaniach, zwykłym i generowanym. Jeśli wywiera na nie inny wpływ, to dlaczego?

3. Przypiesz jeszcze bardziej: jeśli ostatnia liczba w sekwencji jest niezerowa, to co możemy zaoszczędzić?

4. Dodaj parametr $M$ z oryginalnego zadania – maksymalna długość podsekwencji. Zadbaj przy tym, żeby w generowanym programie nie pojawił się żaden dodatkowy `if`.

---

**Zadanie 3.** *Przypominam, że do każdej listy w SKOSie jest jeszcze do zrobienia zadanie dla sprawdzaczki, które ma osobny termin.*