# Лабораторна робота №5. Циклічні конструкції

## 1 Вимоги

### 1.1 Розробник

- Кузнецов Микита Олександрович
- студент групи КІТ-320
- 15-nov-2020

### 1.2 Загальне завдання

Реалізувати програму відповідно до індивідуального завдання за допомогою трьох типів циклів: for, while-do, do-while (отримати три однакових результати).

### 1.3 Індивідуальне завдання

Визначити, чи є задане число досконалим (якщо воно дорівнює сумі своїх дільників). Наприклад, 6 – досконале число, бо 6 = 1 + 2 + 3.

## Опис програми

### 1.4 Функціональне призначення

Програма призначена для визначення досконалого числа, що декларовано в файлі *PerfectNumber.c*.

Результат обчислення зберігається у змінній *resultFor, resultWhile, resultDoWhile*.

Демонстрація отриманих результатів передбачає покрокове виконання програми в режимі налагодження.

### 1.5 Опис логічної структури

За допомогою #define задаємо початкові дані для числа.

Для отримання результату використовується функція main, що знаходиться в *PerfectNumber.c*

### Структура проекту

```
.
├── dist
│   └── PerfectNumber.bin
├── Doxyfile
├── Makefile
├── README.md
└── src
    └── PerfectNumber.c
```

### 1.6 Важливі фрагменти програми

### Початкові дані. Define

```
#define NUM 6 // Заданное число
```

### Обчислення через For

```
int sumFor = 0;
int resultFor;
int tempFor;
for (int i = 1; i < NUM; i++) {
		tempFor = NUM % i;
		if (tempFor == 0) {
			sumFor += i;
		}
	}
	if (sumFor == NUM) {
		resultFor = 0; // 0 - Число совершенное
	} else if (sumFor != NUM) {
		resultFor = 1; // 1 - Число не совершенное
	}
```

### Обчислення через While

```
	int sumWhile = 0;
	int resultWhile;
	int tempWhile;
	int j = 1;
	while (j < NUM) {
		tempWhile = NUM % j;
		if (tempWhile == 0) {
			sumWhile += j;
		}
		j++;
	}
	if (sumWhile == NUM) {
		resultWhile = 0; // 0 - Число совершенное
	} else if (sumWhile != NUM) {
		resultWhile = 1; // 1 - Число не совершенное
	}
```

### Обчислення через Do While

```
	int sumDoWhile = 0;
	int resultDoWhile;
	int tempDoWhile;
	int k = 1;
		do {
		tempDoWhile = NUM % k;
			if (tempDoWhile == 0) {
				sumDoWhile += k;
			}
			k++;
		} while (k < NUM);
		if (sumDoWhile == NUM) {
			resultDoWhile = 0; // 0 - Число совершенное
		}else if (sumDoWhile != NUM) {
			resultDoWhile = 1; // 1 - Число не совершенное
		}
```

## 2 Варіанти використання

Для демонстрації результатів використовується покрокове виконання програми в інтегрованому середовищі *Nemiver*. Нижче наводиться послідовність дій запуску програми у режимі відлагодження.

Крок 1 (див рис. 1). Знаходячись в основної процедурі, досліджуємо стан змінних

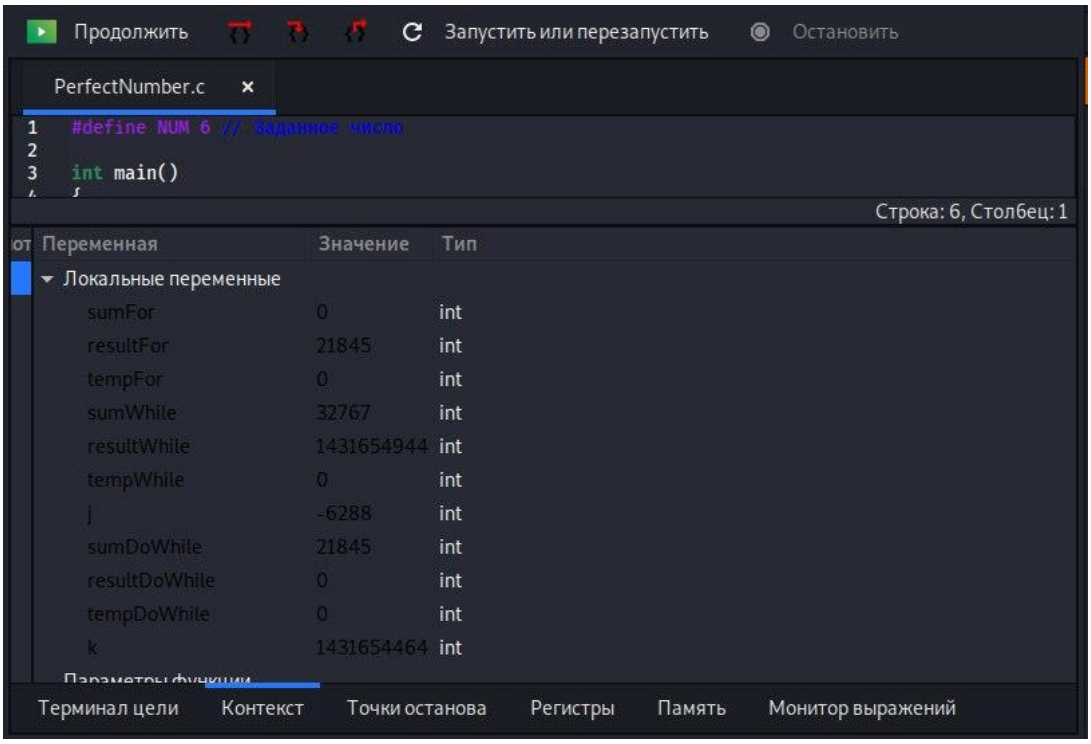


*Рисунок 1 – вікно відлагодження в основної процедурі*

Крок 2. Дослідження стану змінних наприкінці виконання основної функції. Результат зображено на рис. 2, результат обчислення відстані можна побачити у змінній *resultFor*, *resultWhile*, *resultDoWhile*.

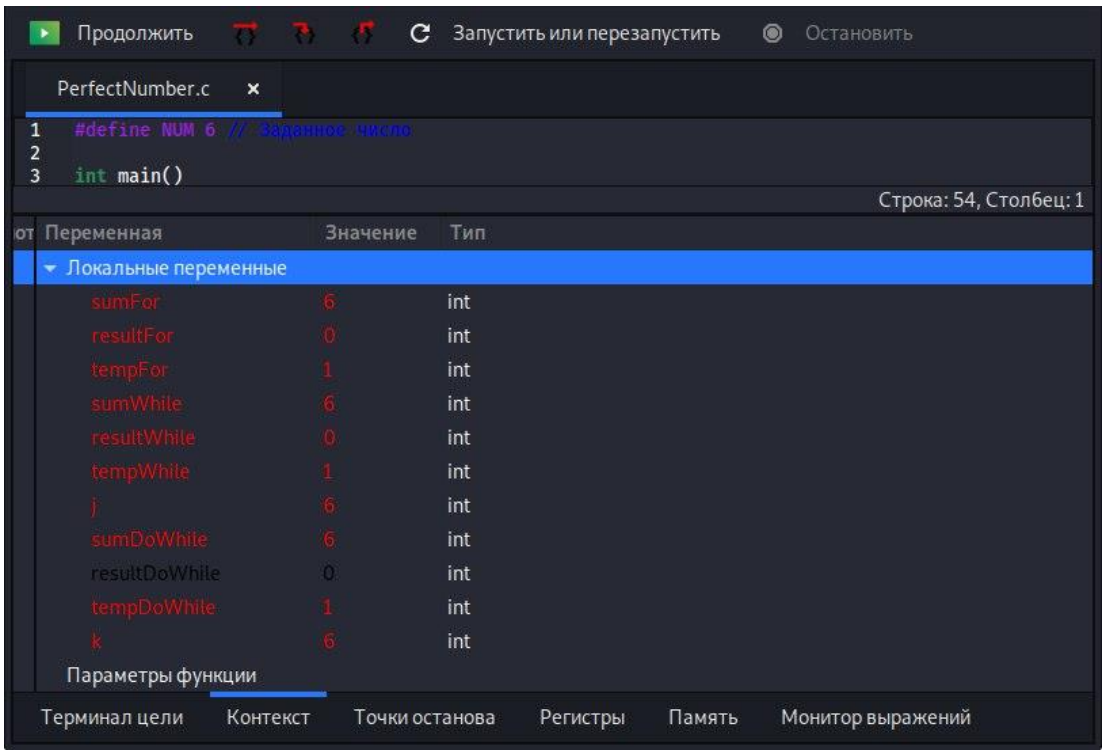


*Рисунок 2 – вікно відлагодження з результатом*

## 3 Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з циклічними конструкціями *for, while, do while.*