

ZADANIA Z JĘZYKA C DLA GRUP 2. I 5.

Informacje uzupełniające - październik/listopad 2025

Parametry opcjonalne funkcji main. W wielu sytuacjach bardzo użyteczna okazuje się możliwość wywołania programu z parametrami opcjonalnymi. Zakładamy, że pewne parametry powinny mieć swoje domyślne wartości, które najprawdopodobniej będą często używane (a zatem nie jest wskazane pytanie o nie przy każdorazowym wywołaniu programu), jednak użytkownik powinien mieć możliwość stosunkowo łatwej ich modyfikacji. Rozwiązaniem przyjętym dla komend systemów typu *Unix* jest podawanie takich parametrów ze znakiem "-", po którym następuje jednoliterowy identyfikator parametru, a dalej (bez odstępu) jego wartość. (Przykładowo, najwyższy poziom optymalizacji kompilowanego programu uzyskujemy wpisując: `gcc -O3`).

Implementację parametrów opcjonalnych zilustrujemy na przykładzie programu obliczającego silnię metodą rekurencyjną lub iteracyjną. Zakładamy, że domyślna jest metoda rekurencyjna (`metoda == 1`), użytkownik może jednak wybrać metodę iteracyjną, wpisując np. `./silnia.out -M2 5`, aby obliczyć $5! = 120$. Funkcja `main` programu może wówczas wyglądać następująco:

```
int main(int narg, char *argv[])
{
    int n,c,metoda;
    double silnia;

    metoda = 1; // Domyślna metoda rekurencyjna //

    while (--narg>0 && (*++argv)[0]!='-') {
        if (c=*++argv[0]) {
            switch (c) {
                case 'M':
                    sscanf(++argv[0],"%d",&metoda);
                    break;
                default:
                    printf("Nieznana opcja: -%c\n",c);
                    return 1;
            }
        }
    }

    // Zakładamy, że <n> jest argumentem obowiązkowym //
    if (narg<1) {
        printf("> ./silnia [-Mmetoda] <n>\n");
        return 2;
    }
    sscanf(argv[0],"%d",&n);
```

```

if (metoda==1) {
    silnia = silnia_rekurencyjnie(n);
}
else if (metoda==2) {
    silnia = silnia_iteracyjnie(n);
}
else {
    printf("Nieznana metoda: %d\n",metoda);
    return 3;
}

// Wydruk wynikow: //
printf("Wynik: %d! = %g\n",n,silnia);
return 0;
}

```

gdzie funkcje

```
double silnia_rekurencyjnie(int);
```

oraz

```
double silnia_iteracyjnie(int);
```

trzeba zdefiniować dodatkowo. Proszę zwrócić uwagę na obsługę błędów wywołania programu poprzez zwracanie wartości 1, 2, lub 3, w zależności od rodzaju błędu.

Program można bardzo łatwo zmodyfikować, aby np. po użyciu argumentu opcjonalnego `-N10000` silnia była obliczana (wybraną metodą) 10000 razy, co powinno umożliwić pomiar czasu trwania obliczeń w praktyce.