# Programowanie i metody numeryczne

Zadania – seria 6.

Całkowanie.

## Zadanie 1. `funcparams` – Funkcje jako argumenty funkcji I.

Napisz zestaw następujących funkcji:

- ```
  std::vector<double> transform(const std::vector<double> & x,
      double(*f)(double))
  ```

  – funkcja ta powinna zwracać wektor, którego kolejne elementy są wartościami funkcji `f` od kolejnych elementów wektora `x`,

- ```
  std::vector<double> transformLess(const std::vector<double> & x, double g,
      std::function<void(double)> f)
  ```

  – funkcja ta powinna zwracać wektor, którego każdy element jest odpowiednim elementem wektora `x`, gdy element ten jest większy od `g`, lub waartością funkcji `f` od tego elementu w przeciwnym przypadku,

- ```
  typedef bool (*greater)(double, double);
  double customMax(const std::vector<double> &v, greater f);
  ```

  – funkcja ta powinna zwracać wartość największego elementu wektora `v`, używając funkcji `f` do porównywania elementów; przyjmij, że $a$ jest większe od $b$, gdy `f(a, b)` ma wartość `true`.

Korzystając z tych szablonów, napisz program `funcparams`, który będzie w dowolny sposób testował ich działanie.

## Zadanie 2. `sampling` – Funkcje jako argumenty funkcji II.

Napisz zestaw następujących funkcji:

- ```
  std::vector<double> sampling1(double from, double to, size_t n,
      double(*f)(double))
  ```

- ```
  std::vector<double> sampling2(double from, double to, size_t n,
      std::function<double(double)> f)
  ```

- ```
  typedef double (*func)(double);
  std::vector<double> sampling3(double from, double to, size_t n, func f)
  ```

- ```
  template<double (*F)(double)>
  std::vector<double> sampling3(double from, double to, size_t n, F f)
  ```

- ```
  template<typename F>
  std::vector<double> sampling3(double from, double to, size_t n, F f)
  ```

Każda z tych funkcji powinna zwracać wektor wartości funkcji `f` obliczonych w równoodległych punktach z przedziału $a, b]$

Korzystając z tych funkcji, napisz program `sampling`, który będzie dla kilku testowych funkcji `f` porównywał czasy działania napisanych przez Ciebie funkcji. Do obliczenia czasu wykonywania kodu możesz użyć pliku nagłówkowego `chrono`.

## * Zadanie 3. `intnum` – Kwadratury interpolacyjne wielomianów.

Niech będzie dana całka oznaczona

$$I = \int_a^b f(x)\,\mathrm{d}x$$

z pewnej funkcji całkowalnej $y = f(x)$.

Napisz zestaw szablonów funkcji obliczających wartość tej całki numerycznie:

- ```
  template<typename F>
  double intRectangular(F f, double a, double b, int n)
  ```

  – metodą prostokątów: całka powinna być przybliżana przez sumę pól $n$ prostokątów, przy czym wysokość prostokąta powinna odpowiadać wartości całkowanej funkcji w środku przedziału,

- ```
  template<typename F>
  double intTrapezoidal(F f, double a, double b, int n)
  ```

  – metodą trapezów: całka powinna być przybliżana przez sumę pól $n$ trapezów,

- ```
  template<typename F>
  double intSimpson(F f, double a, double b, int n)
  ```

  – metodą Simpsona.

Napisz program testowy, sprawdzający działanie tych szablonów dla kilku funkcji podcałkowych (np. $x^2$, $\sin x$).

Następnie, korzystając z tych szablonów, napisz program `intnum`, który przyjmuje jako argumenty wywołania jedno ze słów: `rectangular`, `trapezoidal` lub `Simpson`, określające metodę całkowania, oraz dwie liczby rzeczywiste określające granice całkowania. Program powinien wczytywać ze standardowego wejścia liczby rzeczywiste aż do napotkania znaku końca pliku, następnie konstruować wielomian z tymi liczbami jako współczynnikami i obliczać całkę zadaną metodą numeryczną w zadanych granicach.

*Opracowanie: Bartłomiej Zglinicki.*
  ```