Programowanie Zaawansowane FM i NI
Ćwiczenia 6

Zadanie 1 (tablica alokowana dynamicznie)

Zakoduj dynamiczne tworzenie tablicy `N` komórek typu `int` (`N` podaje użytkownik z klawiatury). Niech użytkownik wypełni tę tablicę kolejnymi liczbami z klawiatury. Na koniec wypisz komórki tablicy i ją skasuj.

W drugim kroku obejmij pętlą nieskończoną cały cykl „tworzenie tablicy – wypełnianie – wypisywanie – kasowanie tablicy". Niech wyjściem z pętli będzie wpisanie przez użytkownika żądanej długości tablicy ≤ 0.

Następnie rozdziel kod. W funkcji `main` pozostaw deklarację pustego wskaźnika `int* pamiec` i samą ramę pętli. Natomiast całość działań dotyczących jednego kroku (tworzenie – wypełnianie – wypisywanie – kasowanie) przekaż do podfunkcji `Zarzadzenie (int* pamiec)`.

Zadanie 2a (Bisection → template)

Zmień funkcję `Bisection` z zadania 4.7 na szablon (`template`), tak aby uniezależnić algorytm od typu zmiennych, na których pracuje. Funkcję `myfun` też przerób na szablon.

Następnie powiel wywołanie `Bisection` w funkcji `main` tak, aby dodatkowo algorytm poszukał tego samego w dziedzinie liczb typu `float`. Uwaga: stałą liczbową typu `float` można uzyskać np. tak: `1.23f`.

Teraz umyślnie popsuj na chwilę kod, wywołując `Bisection` tak, aby argumentami były napisy (np. "abc") – i zaobserwuj monity kompilatora.

Zadanie 2b (Bisection → lambda)

Zmodyfikuj kod bisekcji w ten sposób, aby funkcji `main` wywołać `Bisection` poprzez wyrażenie lambda, służące za analizowaną funkcję f(x) = exp(–x) – x. W tym celu:

[1] w funkcji `main` utwórz obiekt z dedukcją typu przez `auto`, któremu przypiszesz Twoje wyrażenie lambda.

[2] Szablon `myfun` możesz usunąć z kodu.

[3] samo `Bisection` pozostaw jako szablon. Z tym, że ponieważ lambda ma inny typ, niż dane, to zdefiniuj dwa typy uogólnione, np.:

```
template<typename FunType, typename DataType>
```

[4] W funkcji `main` wywołaj `Bisection` , podając Twój obiekt z lambdą.

Poeksperymentuj. Możesz np.:

- miejsce deklaracji lambdy przenieść od razu do miejsca wywołania `Bisection` .
- W argumencie wejścia `Bisection` , służącym do przekazu obiektu lambda, możesz zmienić z typu uogólnionego na obiekt o typie dedukowanym za pomocą `auto` (jak też `auto&`) .
- Możesz zadeklarować lambdę, a następnie zadeklarować wskaźnik funkcyjny, której tę

lambdę przypiszesz (Takie konwersje działają, o ile lambda nie ma cechy *capture*). W argumencie wejścia `Bisection` potrzebujesz albo utrzymać dedukcję `auto`, albo powrócić do wskaźnika funkcyjnego.

Zadanie 3 (`template, lambda`)

Rozważ kod dla funkcji `Analiza` z zadania 4.2 (lub jego usprawnienie w zad. 4.9). Przepraw go tak, aby:

- funkcja `Analiza` stała się szablonem, w którym typ danych liczbowych uogólniony jest do `T`, a typ obiektu przyjmującego funkcję zawierającą funkcję matematyczną – do `F`.
- funkcja matematyczna została zmieniona z funkcji `double f (double x)` do wyrażenia Lambda, przechowywanego w środku `main`. Utwórz dwa warianty takiego wyrażenia, o nazwach – odpowiednio – `MyLamF` (pracującego na liczbach typu `float`) oraz `MyLamD` (na `double`).
- w `main` dwukrotnie wywołaj `Analiza`, dla wariantów `float` i `double`. Zadbaj, aby w danym wariancie wszystkie argumenty wejścia były jednolitego typu (np. stałą liczbową typu `float` otrzymasz przez np. `1.2f` albo `(float)1.23`.

Zadanie 4 (zmienne `static` w funkcji)

W tym kodzie zaimplementowana jest funkcja licząca N-ty wyraz ciągu Fibonacciego wprost i przez rekurencję.

Dodaj funkcję `Fibonacci_NewIteration`, która będzie posiadać 3 zmienne statyczne do przechowywania aktualnych wartości 3 ostatnich wyrazów ciągu. Pojedyncze wywołanie tej funkcji ma spowodować aktualizację tych zmiennych o kolejny (następny) wyraz.

W funkcji `main` dodaj trzeci wariant liczenia wyrazu ciągu i pomiar czasu w tej metodzie. Sprawdź działanie dla N = 40.

Zadanie 5 (zmienne `static` w funkcji)

Rozwinięcie e^x wokół 0 ma postać: $\exp(x) = \sum_{n=0}^{N} a_n$, gdzie $a_n = \frac{x^n}{n!}$, przy czym zachodzi: $a_n = a_{n-1} \cdot \frac{x}{n}$. W odróżnieniu od podejścia rekurencyjnego w serii 2, do obliczenia e^x zastosuj podejście zmiennych statycznych. Zakoduj funkcję: `double expStat_NewIter (double x)`, która będzie posiadać 3 zmienne statyczne:

- `double Nstep`, pamiętającą nr kroku
- `double a_n`, pamiętającą wartość n-tego wyrazu
- `double sum`, pamiętającą sumę szeregu z poprzedniego wywołania funkcji.

Po aktualizacji tych zmiennych, niech `expStat_NewIter` zwraca wartość sumy szeregu po aktualnym kroku.

W funkcji `main` w pętli wywołaj 10× `expStat_NewIter` dla $x = 1$, za każdym krokiem wypisując jej wynik na ekran.

Zadanie 6 Profil Voigta

Jeżeli zjawisko fizyczne cechuje rozkład Lorentza $L(x)$, ale rozdzielczość detektora rozmywa każdą wartość x zgodnie z funkcją Gaussa $G(x)$, to w wyniku obserwujemy tzw. profil Voigta $V(x)$, będący splotem tych dwóch funkcji:

$$V(x;\ \sigma,\gamma) = G(x;\ \sigma) \circ L(x;\ \gamma) = \int_{-\infty}^{\infty} L(x-x';\ \gamma) \cdot G(x';\sigma)\ dx'$$

gdzie:

$$G(x;\ \sigma) = \frac{1}{\sigma\sqrt{2\pi}}\ \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

$$L(x;\ \gamma) = \frac{\gamma}{\pi(x^2\ +\ \gamma^2)}$$

oraz σ i γ – są parametrami tych funkcji. Napisz wpierw odpowiednie funkcje Lorentza i Gaussa:

```
double Lorentz (double x, double gamma);
double Gauss   (double x, double sigma);
```

oraz funkcję, która dla argumentu x i ustalonych parametrów σ i γ zwraca wartość splotu powyższych funkcji:

```
double Voigt   (double x, double sigma, double gamma);
```

Splot jest wykonywany numerycznie i aby to uczynić, koncepcyjnie trzeba przejść z całki na sumę:

$$\int_{-\infty}^{\infty} L(x-x';\ \gamma) \cdot G(x';\sigma)\ dx' \ \rightarrow \sum_{x'\ =\ -5\sigma}^{+5\sigma} L(x-x';\ \gamma) \cdot G(x';\sigma)\ \Delta x'\ ,$$

gdzie zakres $x' \in [-5\sigma, 5\sigma]$ i iteracja co $\Delta x' = 0.01$ dość dobrze przybliżą całkę z funkcji Gaussa po $x' \in \mathbb{R}$.

W funkcji `main` wypisz wartości splotu w przedziale $x \in [-10, 10]$ dla parametrów $\sigma = 1$ i $\gamma = 1$. Następnie, mając powyższy rezultat:

– przekształć wszystkie powyższe funkcje w szablony funkcji (`template`) oraz

– funkcje `Lorentz` oraz `Gauss` przekształć w wyrażenia lambda, deklarowane na poziomie szablonu `Voigt`. (Efektywnie jest zadeklarować je jeszcze przed pętlą po x, tak aby nie były one tworzone i kasowane w każdym kroku pętli.)