Języki programowania

należy napisać program umożliwiający całkowanie różnych funkcji numeryczne dwoma metodami (pudełkową i trapezową).

1. **Klasy reprezentujące funkcje** (1.5 pkt – za przetestowany kod)

Tworzymy klasę abstrakcyjną (ATD). Jest to klasa reprezentująca dowolną funkcję.

Klasa będzie się nazywać Function. Jej plik nagłówkowy powinien zawierać metodę Calculate(double x) jako czysto wirtualną, tzn w pliku Function.h:

```
virtual double Calculate(double x) = 0;
```

Wskazówka: wszystkie metody czysto wirtualne muszą istnieć w klasach pochodnych.

Przypomnienie: klasa, która ma co najmniej jedną funkcję czysto wirtualną nazywamy **klasą abstrakcyjną**.

Tworzymy dwie klasy pochodne: FunctionSin oraz FunctionPolynomial (w tym samym pliku co klasa bazowa!)

(1) FunctionSin reprezentuje funkcję: $f(x) = a \cdot \sin(b \cdot x)$

(2) FunctionPolynomial reprezentuje wielomian: $f(x) = ax^2 + bx + c$

W obu klasach należy przeciążyć metodę Calculate by odpowiadała właściwej funkcji oraz odpowiednie konstruktory ustawiające parametry funkcji (a, b, c…).

1. **Całkowanie** (2 pkt – za przetestowany kod)

Należy napisać dwie funkcje globalne które przyjmują referencję (bądź wskaźnik) na Function :

```
double IntegrateRectangle(const Function& f, double a, double b, int n);
double IntegrateTrapezoid (const Function& f, double a, double b, int n);
```

Szczegóły metod całkowania:

- n – ilość kroków (dokładność całkowania)
- krok całkowania numerycznego na przedziale (a, b):

$$\Delta x = \frac{b-a}{n}$$

- całkowanie **metodą prostokątów (pudełkową)**:

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n-1} f(x_i)\,\Delta x$$

- całkowanie **metodą trapezów**:

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]$$

Piszemy program testujący działanie dwóch wyżej wymienionych klas

- Tworzymy w funkcji main obiekty:
  - klasy FunctionSin: f(x) = sin(x)
  - klasy FunctionPolynomial $f(x) = 5x^2 + 2x + 1$
- Należy wypisać na ekranie całki liczone metodą prostokątów dla obu funkcji na przedziale 0-Pi() dla sinusa oraz 0-2 dla wielomianu dla 10 kroków.
- Należy wypisać na ekranie całki liczone metodą trapezową dla obu funkcji na przedziale 0-Pi() dla sinusa oraz 0-2 dla wielomianu dla dziesięciu kroków.

2. **Porównanie dokładności (0.5 pkt)**

- Zwiększyć liczbę kroków do 100
- Sprawdzić, czy wyniki obu metod się do siebie zbliżają

Na koniec należy wykonany program umieścić w serwisie github (szczegółowa instrukcja na następnej stronie).

- 1 pkt

GIT

Git to **system kontroli wersji**, który służy do śledzenia zmian w plikach (najczęściej w kodzie źródłowym) oraz do pracy zespołowej nad projektami.
Dzięki Gitowi można:

- zapisywać kolejne wersje projektu (historię zmian),
- cofać się do wcześniejszych wersji plików,
- pracować nad tym samym projektem równocześnie z innymi osobami,
- bezpiecznie przechowywać kod w repozytorium (lokalnym lub zdalnym, np. na GitHubie).

Git działa lokalnie na komputerze użytkownika, a serwisy takie jak **GitHub**, **GitLab** czy **Bitbucket** umożliwiają przechowywanie repozytoriów w sieci i łatwą współpracę zespołową.

Instrukcja:
Zadanie polega na:

- założeniu konta w serwisie hostującym repozytoria Git (np. GitHub),
- utworzeniu własnego repozytorium.

Krok 1 – utworzenie konta w serwisie

- Należy założyć konto w wybranym serwisie hostującym repozytoria (jeśli nie zostało jeszcze założone),

Krok 2 – wygenerowanie klucza SSH
W terminalu należy wygenerować klucz SSH, np. poleceniem:

```
ssh-keygen -t ed25519 -C "twoj_email@github.com"
```

Polecenie to **generuje dwa pliki**:

- klucz publiczny,
- klucz prywatny.

Klucz publiczny

- zawartość klucza publicznego należy skopiować do serwisu hostującego repozytoria,
- miejsce wklejenia zależy od serwisu (np. na GitHubie:
  Profil użytkownika → Settings → SSH and GPG keys).

Klucz prywatny

- klucz prywatny **zostaje na komputerze użytkownika**,
- **NIE WOLNO** go nikomu udostępniać – umożliwia on podszywanie się pod właściciela konta.

Krok 3 – dodanie klucza do agenta SSH
Należy wykonać w terminalu polecenie:

```
ssh-add
```

oraz ustawić odpowiednie prawa dostępu do pliku z kluczem prywatnym:

```
chmod 600 nazwa_pliku_klucza_prywatnego
```

Krok 4 – utworzenie repozytorium
W serwisie github należy:

- utworzyć nowe repozytorium (dla dzisiejszych zajęć: prywatne, dla projektów najlepiej publiczne)

Krok 5 – sklonowanie repozytorium na komputer
Repozytorium należy skopiować na lokalny komputer poleceniem:

```
git clone "adres_naszego_repozytorium"
```

Po wykonaniu powyższych kroków repozytorium jest gotowe do pracy. Do wypychania zmian możemy użyć komendy git-gui. To graficzny interfejs pozwalający nam zatwierdzać zmiany i wysyłać do zdalnego repozytorium bez zabawy komendami.