Języki programowania
należy napisać program umożliwiający generowanie liczb pseudolosowych z trzech różnych rozkładów (jednorodnego, Gaussa oraz Poissona).

Tworzymy klasę abstrakcyjną (ATD) – 1 p. Jest to klasa do generowania liczb losowych .
Klasa będzie się nazywać Losuj. Jej plik nagłówkowy powinien zawierać metodę Exec() jako czysto wirtualną, tzn w pliku Losuj.h:

```
virtual double Exec() = 0;
```

Ponadto tworzymy konstruktor domyślny – ustawia ziarno z zegara systemowego: srand(time(NULL))
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: Jednorodny i Gauss
(1) Tworzymy klasę Jednorodny, która dziedziczy z klasy Losuj. Ponadto posiada:

- pola double fMin i double fMax – przedział, z którego będą losowane liczby
- konstruktor domyślny który ustawia fMin na 0 i fMax na 1
- Konstruktor z parametrami ustawiający pola fMin i fMax; srand ustawiamy analogicznie jak w konstruktorze bez parametrów.
- metoda double Exec() - metoda wirtualna, zwraca liczbę pseudolosową z rozkładu jednorodnego z przedziału <fMin;fMax)

(2) Następnie tworzymy klasę Gauss, która dziedziczy z klasy Losuj. Powinna ona zawierać:

- pola typu double fMean, double fSigma, które przechowują parametry rozkładu Gaussa: średnią i odchylenie standardowe.
- konstruktor z parametrami ustawiający wszystkie pola
- metodę double Exec() - zwracającą liczbę pseudolosową z rozkładu Gaussa o zadanej średniej i odchyleniu standardowym (szczegóły implementacji patrz **Uwaga1!**)

Zestaw bibliotek niezbędnych do stworzenia generatora:
cstdlib, ctime, cmath
Należy użyć funkcji rand() - zwracającej liczbę całkowitą pseudolosową z rozkładu <0;RAND_MAX), gdzie RAND_MAX jest wielką liczbą - stałą zdefiniowaną w bibliotece standardowej.

Piszemy program testujący działanie dwóch wyżej wymienionych klas – 2.5 p.
Tworzymy w funkcji main obiekt klasy Jednorodny (np. uni1(-2,3);) z ustawionymi wartościami od -2 do 3. Losujemy kilka liczb pseudolosowych i wypisujemy na ekran.
Tworzymy w funkcji main obiekt klasy Gauss (np. gaus1(100,5);) z ustawioną średnią na 100 i odchyleniem 5. Losujemy kilka liczb pseudolosowych.
Używając operacji przekierowania strumienia należy zapisać wypisywane liczby do plików uniform.txt i gauss.txt.

GIT – 1 p.
Na koniec należy wykonany program umieścić w serwisie github (szczegółowa instrukcja na ostatniej stronie).

Klasa do generowania liczb z rozkładu Poissona – 0.5 p. należy dodać klasę Poisson (czytaj **Uwaga2!**), zachowującą się podobnie do poprzedniej, wywołać na niej metodę Exec().

Uwaga 1!
Algorytm do generowania liczb pseudolosowych z rozkładu Gaussa (metoda Box-Muller'a):

- U, V – dane liczby pseudolosowe z rozkładu jednorodnego <0;1)
- Liczba losowa z rozkładu Gausa o średniej μi odchyleniu standardowym σdana jest wtedy wzorem:

$$x = \mu + \sigma\sqrt{-2\ln(U)}\cos(2\pi V)$$

Uwaga 2!
Algorytm do generowania liczb pseudolosowych z rozkładu Poissona o średniej λ:

- $x_0, x_1, \ldots$- ciąg liczb pseudolosowych z rozkładu jednorodnego
- Jeżeli $x_0 x_1 \ldots x_k < \mathrm{e}^{-\lambda}$, to k jest liczbą z rozkładu Poissona.

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.