# ZAAWANSOWANE PROGRAMOWANIE KOMPUTEROWE WNE (2022)

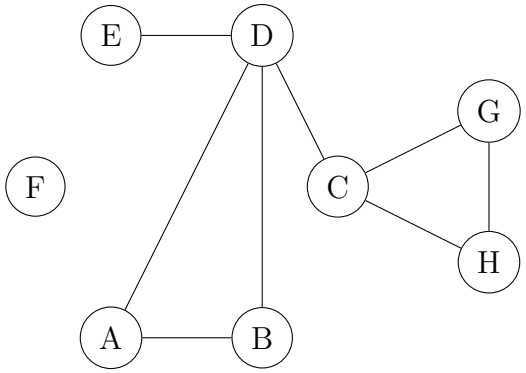
KRZYSZTOF ZIEMIAŃSKI

## 13. Grafy

Graf to obiekt, który składa się ze

- *zbioru wierzchołków* $V$
- *zbioru krawędzi*: każda para wierzchołków albo jest połączona krawędzią albo nie jest.

Przykładowy graf:




Tematem tych zajęć będzie implementacja klasy `Graf`, której obiektami będą grafy. Wierzchołki będą obiektami pewnego wybranego typu; zależy na na tym, aby nie ograniczać się np. do typu `int` (tak, jak to robiliśmy w przypadku list). Stworzymy więc **wzorzec** klas, a nie pojedynczą klasę. Przykładem takiego wzorca jest `vector`: w C++ każdy wektor musi mieć przypisany konkretny typ elementów, które się w nim znajdują.

13.1. **Wzorce.** Prostym przykładem funkcji jest funkcja zwracająca większy z dwóch parametrów:

```
int max(int a, int b) {
    if(a>b)
        return a;
    else
        return b;
}
```

Ta funkcja będzie działać dla parametrów typu `int`, ale jeśli będziemy potrzebować funkcji zwracającej większą z dwóch liczb typu `double` albo napisów (`string`) musimy napisać osobną funkcję dla każdego typu, która będzie prawie identyczna. Wzorce to sposób na zmuszenie kompilatora, aby sam sobie stworzył taką funkcję. Poniższy kod zawiera definicję odpowiedniego wzorca:

```
template <class T>
T max(T a, T b) {
```

```
    if(a>b)
        return a;
    else
        return b;
}
```

Deklaracja `template <class T>` oznacza, że bezpośrednio następująca funkcja (lub klasa) jest wzorcem zależnym od pewnego typu `T`. Teraz możemy wywoływać funkcję `max` dla parametrów dowolnych typów — pod warunkiem, że oba parametry są tego samego typu i są porównywalne (tj. działa dla nich operator `>`). Kompilator sam stworzy potrzebną funkcję w razie potrzeby.

Można też tworzyć klasy parametryzowane pewnym typem. Przykładowa definicja wzorca klasy `Punkt`, gdzie parametrem jest typ współrzędnej punktu:

```
template <class T>
class Punkt {
public:
    Punkt(T _x, T _y): x(_x), y(_y) {}
    void przesun(T dx, T dy) { x+=dx; y+=dy; }
    void drukuj();
    T x;
    T y;
};

template <class T>
void Punkt<T>::drukuj() {
    cout << "(" << x << "," << y << ")";
}
```

Teraz możemy wybierać, jakiego typu będą współrzędne punktu, np. deklaracje

```
Punkt<int> p;
Punkt<double> q;
```

stworzą punkty o różnych typach współrzędnych.

Uwaga: jeśli definiujemy metody "poza" klasą, musimy powtórzyć deklarację `template` przed definiowaną funkcją.

13.2. **Wzorzec `Graf`.** Klasa `Graf<T>` będzie miała następujące metody:

- `Graf();` konstruktor, tworzy pusty graf (bez wierzchołków),
- `void dodaj(T nowy);` dodaje nowy wierzchołek o podanej nazwie,
- `void polacz(T v1, T v2);` dodaje połączenie pomiędzy podanymi wierzchołkami,
- `bool zawiera(T v);` sprawdza, czy graf zawiera dany wierzchołek,
- `bool polaczone(T v1, T v2);` sprawdza, czy wierzchołki są połączone,
- `vector<T> listaWierzcholkow();` zwraca listę wierzchołków grafu,

- `vector<T> listaSasiadow(T v);` zwraca listę wierzchołków połączonych z podanym wierzchołkiem,
- `void polaczLosowo(double p);` każda para wierzchołków zostaje połączona z prawdopodobieństwem `p`.

Każdy wierzchołek grafu posiada dwa unikalne identyfikatory:

- publiczny (zwany dalej nazwą) — typu `T`, używany przez użytkownika klasy `Graf<T>`,
- wewnętrzny (zwany dalej numerem) — typu `int`, używany tylko w metodach klasy. Dbamy o to, aby były to kolejne numery 0,1,. . . ,n-1.

W tablicy `nazwy` znajduje się przyporządkowanie numerom wierzchołków ich nazw, czyli wierzchołek o numerze `k` ma identyfikator publiczny `t[k]`. Do tłumaczenia nazw na numery służy funkcja `indeks`, która zwraca `-1` jeśli wierzchołek o podanej nazwie nie istnieje.

Połączenia pomiędzy wierzchołkami są zapamiętywane w dwuwymiarowej tablicy `tab`: `tab[k][l]` jest prawdą, gdy istnieje połaczenie pomiędzy wierzchołkami o numerach `k` i `l` (a fałszem jeśli takiego połączenia nie ma).

13.3. **Zadania.** Należy uzupełnić klasę `Graf` o następujące metody:

(1) `void usunPolaczenie(T v1, T v2);` usuwa połączenie pomiedzy podanymi wierzchołkami; jeśli takich nie ma, nic się nie dzieje.

(2) `int liczbaKrawedzi();` zwraca liczbę wszystkich krawędzi w grafie.

(3) `void usunWierzcholek(T v);` usuwa wierzchołek o podanej nazwie (o ile istnieje).

(4) `bool spojny();` zwraca `true` jeśli graf jest spójny, tzn. z każde dwa wierzchołki są połączne za pomocą krawędzi (być może nie bezpośrednio).