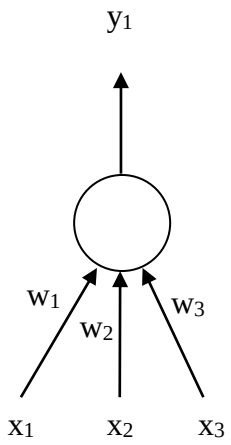
Semestrální úloha č. 1 z předmětu
Programování a modelování

Naimplementujte objektově McCulloch–Pittsův model neuronu (bez učení). Parametrem konstruktoru nechť je vektor vah. Definujte společného předka a několik potomků implementujících alespoň dvě různé přenosové funkce. Implementaci ověřte v hlavním programu na zadaných hodnotách vstupních vektorů – vypište hodnotu výstupu neuronu. Součástí práce bude diagram tříd v UML.

Výstupní funkce neuronu: $y = \Phi(x_1 w_1 + \cdots x_n w_n + \Theta)$

Φ ... přenosová funkce neuronu (skoková funkce, sigmoida, ..)
Θ ... práh



Semestrální úloha č. 2 z předmětu
Programování a modelování

Naimplementujte objektový model fuzzy množiny. Implementujte alespoň dva typy fuzzy množin s různými funkcemi příslušnosti (lichoběžník, trojúhelník, ...). Navrhněte vhodné atributy a metody, povinně implementujte metodu vracející míru příslušnosti prvku (reálného čísla) k množině. Implementaci ověřte v hlavním programu. Součástí práce bude diagram tříd v UML.

(Návod: Definujte abstraktní třídu jako společného předka, od ní odvoďte třídu pro každý typ fuzzy množiny. Využijte virtuální metody.)

Semestrální úloha č. 3 z předmětu
Programování a modelování

Naimplementujte objektově komplexní kalkulačku. Nadefinujte třídu, která reprezentuje komplexní číslo a metody implementující základní aritmetické operace (sčítání, odčítání, násobení, dělení) – metody budou vracet nový objekt - výsledek (komplexní číslo), dále metody pro zjištění velikosti a výpočet parametrů goniometrického tvaru. Nemusíte implementovat přetěžování operátorů. Součástí práce bude diagram tříd v UML.

Semestrální úloha č. 4 z předmětu
Programování a modelování

Naprogramujte textovou konverzační hru, která se ptá hráče na vlastnost zvířat a snaží se podle vlastnosti uhodnout zvíře, které si hráč myslí. Pro jednoduchost uvažujte pouze 1 vlastnost u každého zvířete. Znalostní bázi implementujte dle vlastní úvahy, můžete využít knihovnu STL. Prémiové body získáte za uložení/načtení databáze do/z textového souboru.

Příklad: (P je počítač, H je hráč)

```
P: Mysli se zvíře.
P: Bručí?
H: ano
P: Tak je to medvěd!
P: Chceš hrát dál?
H: ano
P: Mysli se zvíře.
P: Bručí?
H: ne
P: Toto zvíře neznám. Co je to?
H: Pták
P: Co dělá pták?
H: Létá
P: Tedy pták létá
P: Chceš hrát dál?
H: ano
P: Mysli se zvíře.
P: Bručí?
H: ne
P: Létá?
H: ano
P: Tak je to pták!
P: Chceš hrát dál?
atd.
```

Semestrální úloha č. 5 z předmětu
Programování a modelování

Naimplementujte třídu pro reprezentaci lomené čáry v rovině o maximálně 50 bodech. Navrhněte vhodné atributy a metody, povinně implementujte metody celkovou délku čáry a počet segmentů a délku *i*-tého segmentu. Implementaci ověřte v hlavním programu. Součástí práce bude diagram tříd v UML.

Semestrální úloha č. 6 z předmětu Programování a modelování

Naimplementujte třídu pro reprezentaci trojúhelníku v rovině. Navrhněte vhodné atributy a metody, povinně implementujte metody vracející obsah a obvod trojúhelníku a posouvající každý bod. Při konstrukci ověřte, zda zadané body opravdu tvoří trojúhelník. Implementaci ověřte v hlavním programu. Součástí práce bude diagram tříd v UML.

Semestrální úloha č. 7 z předmětu
Programování a modelování

Řízení vjezdu

Navrhněte objektový model aplikace, která simuluje řízení otevírání vjezdové závory podle čísla karty přečtené čtečkou. K dispozici máte dvě hotové třídy: `CardReader`, jejíž metody vrací číslo přečtené karty a označení čidla, které kartu přečetlo, dále `GateControl`, sloužící k řízení každé jednotlivé závory a zobrazení hlášení na displeji (tj. pro každou bránu bude v aplikaci jedna instance).
Navrhněte třídu, která implementuje databázi karet (můžete využít třídy z STL) a rozhodování, zda karta má právo k otevření závory. Navrhněte metody, které vkládají do databáze záznamy o právech otevření bran.

Nakreslete diagram tříd v UML.

Čísla čidel a čísla bran jsou v rozsahu 1 – 10, čísla karet v rozsahu 300 000 – 300 500.

Třídy simulují činnost jednotlivých částí systému.

Metoda `CardReader::is_Read_any_Card()` vrací true, pokud byla přečtena nějaká karta ke čtečce u nějakého stojanu (s pravděpodobností 50% vrací true). Metoda `CardReader::get_Card_and_Reader(int &card, int &gate)` vrací číslo přečtené karty a číslo brány, ke které byla karta přiložena (opět náhodně vygenerované). Správné použití funkcí je, že program se cyklicky dotazuje, zda byla přečtena karta a pak si vyzvedne číslo karty a stojanu.

Metoda `GateControl::open_Gate(int gate)` simuluje otevření závory u brány č. gate. Napíše na obrazovku text, že je posílán příkaz k otevření příslušné brány. Metoda `GateControl::display_message(std::string &m)` simuluje zaslání textu na informační panel u brány (vypíše text na obrazovku). Je na Vás, jak tuto funkci využijete, např. k zobrazení textu „Můžete vjet“, „Vaše karta není v databázi“ apod.