Laboratorium 8 Pomoc. Funkcje i klasy zaprzyjaźnione

Cel laboratorium:
Zapoznanie z wykorzystaniem funkcji i klas zaprzyjaźnionych w programowaniu obiektowym

- **Funkcja zaprzyjaźniona** z klasą to taka, która nie jest składnikiem klasy, a ma dostęp do wszystkich jej elementów
- Deklaracja przyjaźni wewnątrz klasy:

friend <deklaracja przyjaciela>;

- Deklaracja przyjaciela:
  - prototyp funkcji globalnej
  - prototyp metody ze zdefiniowanej wcześniej klasy
  - nazwa zadeklarowanej wcześniej klasy
- Klasy i funkcje zadeklarowane jako **przyjaciele** (**friends**) mają dostęp do prywatnych i chronionych pól i funkcji klasy
- Deklarację przyjaźni składa **klasa** (nie funkcja) słowem **friend**
- Miejsce deklaracji przyjaźni (public, private, protected) **nie jest istotne**
- **Definicja funkcji** może być poza lub wewnątrz klasy
- Do funkcji zaprzyjaźnionej trzeba przesłać **wskaźnik** lub **referencję** obiektu (funkcja **nie ma dostępu do this**)
- Przyjaźń pomiędzy klasami jest **jednostronna** (nie jest automatycznie wzajemna) **i nieprzechodnia**: jeżeli klasa A jest przyjacielem klasy B, która jest przyjacielem klasy C to niekoniecznie B jest przyjacielem A i A nie jest przyjacielem C. Przyjaźń nie jest **dziedziczna.**
- Klasa ma **tylko tych przyjaciół**, których sama sobie zadeklaruje

- Idea funkcji zaprzyjaźnionych **łamie właściwość programowania obiektowego – hermetyzację!!! ale...rozszerza publiczny interfejs klasy,** która decyduje o tym komu da dostęp do wszystkich swoich składowych

Klasy.h

```
#ifndef KLASY_H
#define KLASY_H
#include <iostream>
using namespace std;
class klasa1;     //deklaracja zapowiadająca

class klasa2      //klasa2 z jedną metodą przyjacielem klasy1
{private:
    std::string bank;
    int kwota;
 public:
      klasa2();
      ~klasa2();
      void wyswietl();
      void metoda(klasa1 &ref);     //metoda przyjaciel klasy1
};

class klasa3      //klasa3 z wszystkimi metodami przyjacielami klasy1
{private: std::string dane;
 public:
      klasa3();
      ~klasa3();
      void metoda1(klasa1 *wsk);//metoda przyjaciel
      void metoda2(klasa1 *wsk);//metoda przyjaciel
      void metoda3(klasa1 *wsk);//metoda przyjaciel
};

class klasa1      //klasa1 deklarujaca przyjazn
{private:
    std::string login,haslo;
 public:
      klasa1();
      ~klasa1();
      void wyswietl();
    friend void przyjaciel1(klasa1 *wsk);     //P1:globalna funkcja
    friend void klasa2::metoda(klasa1 &ref); //P2:metoda innej klasy
    friend  class klasa3;                     //P3:inna klasa
};
#endif // KLASY_H
```

Klasy.cpp

```
#include "klasy.h" // class's header file
#include <iostream>
//metody klasy2
klasa2::klasa2()
{ bank="PKO SA";
  kwota=1000000;}
klasa2::~klasa2()
{}
void klasa2::wyswietl()
{ cout<<"bank:"<<this->bank<<endl;
  cout<<"kwota:"<<this->kwota<<endl;
     }
//przyjaciel=metoda klasy; parametr: referencja do obiektu
void klasa2::metoda(klasa1 & ref)
{    cout<<"bank:"<<this->bank<<" kwota:"<<this->kwota<<endl;
     cout<<"login:"<<ref.login<<" haslo:"<<ref.haslo<<endl;
     }
```

```
//metody klasy3
klasa3::klasa3()
{ dane="serwis PL";}
klasa3::~klasa3()
{}
void klasa3::metoda1(klasa1 *wsk)
{cout<<"Dostep w metodzie 1 "<<this->dane<<endl;
 cout<<"login:"<<wsk->login<<" haslo:"<<wsk->haslo<<endl;
     }
void klasa3::metoda2(klasa1 *wsk)
{cout<<"Dostep w metodzie 2  "<<this->dane<<endl;
 cout<<"login:"<<wsk->login<<" haslo:"<<wsk->haslo<<endl;
     }
void klasa3::metoda3(klasa1 *wsk)
{cout<<"Dostep w metodzie 3  "<<this->dane<<endl;
 cout<<"login:"<<wsk->login<<" haslo:"<<wsk->haslo<<endl;
     }
//metody klasy1
klasa1::klasa1()
{login="kwinto"; haslo="vabank"; }
klasa1::~klasa1()
{}
void klasa1::wyswietl()
{cout<<"login:"<<this->login<<" haslo:"<<this->haslo<<endl;
     }
//przyjaciel=f. globalna; parametr: wskaznik do obiektu
void przyjaciel1(klasa1 *wsk)
{cout<<"login:"<<wsk->login<<" haslo:"<<wsk->haslo<<endl;
     }
```

main.cpp

```
#include <cstdlib>
#include <iostream>
#include "klasy.h"
using namespace std;

int main(int argc, char *argv[])
{ klasa1 obiekt1;
  klasa2 obiekt2;
  klasa3 obiekt3;
cout<<endl<<"dostęp do składowych klasy1 przez przyjaciela -funkcję
globalna"<<endl;
 przyjaciel1(&obiekt1);//funkcja przyjaciela
cout<<endl<<"dostęp do składowych klasy1 przez przyjaciela - 1 metode innej
klasy"<<endl;
  obiekt2.metoda(obiekt1);//metoda przyjaciel
cout<<endl<<"dostęp do składowych klasy1 przez przyjaciela -metody innej
klasy"<<endl;
  obiekt3.metoda1(&obiekt1);// klasa przyjaciel
  obiekt3.metoda2(&obiekt1);
  obiekt3.metoda3(&obiekt1);

obiekt1.~klasa1();
obiekt2.~klasa2();
obiekt3.~klasa3();
    system("PAUSE");
    return EXIT_SUCCESS;
}
```