Laboratorium Podstaw Informatyki

1. Przebieg ćwiczenia laboratoryjnego

Abstrakcja klasy

Klasa abstrakcyjna w języku C++ to klasa, która nie reprezentuje żadnego typu obiektu, a ponadto nie możliwe jest stworzenie obiektu należącego do klasy abstrakcyjnej. Mechanizm abstrakcji klasy jest rozszerzeniem mechanizmu wirtualności. Do tej pory tworzyliśmy klasy które w zamyśle niczego nie reprezentowały (np. klasa figura), posiadały funkcje wirtualne, a mimo to możliwe było stworzenie obiektu takiej klasy. W rzeczywistym świecie taka sytuacja nie występuje. Nie można stworzyć samochodu, można natomiast zbudować konkretny model np. Ford Fiesta. Pojazd ten należy do klasy samochodów (odziedziczył po niej wiele cech), ale należy do klasy samochodów Ford Fiesta. Aby w języku C++ stworzyć klasę abstrakcyjną należy przynajmniej jedną z jej metod zdefiniować jako czysto wirtualną. Przykład klasy abstrakcyjnej:

```
class figure
{
 private:
   string name;
 public:
   figure()
    {
      name = "empty";
    }
   void setName(string arg);
   virtual void Draw(void) = 0;      //funkcja czysto wirtualna
};
```

W tym przykładzie funkcja Draw jest funkcją czysto wirtualną. Oznacza się to poprzez przypisanie do niej wartości zero. Jeżeli klasa posiada przynajmniej jedną funkcję czysto wirtualną, to jest to klasa abstrakcyjna! Z takiej klasy nie można stworzyć obiektu. Instrukcja: figure obiekt; spowoduje błąd kompilatora. Zatem każda sytuacja która wymaga stworzenia obiektu klasy abstrakcyjnej (nawet chwilowego) traktowana jest jako błąd.

Nie możemy zatem:

- Stworzyć funkcji która pobiera argumenty klasy abstrakcyjnej przez wartość:
  void funkcja(figura arg);

- Stworzyć funkcji która zwraca obiekty należące do klasy abstrakcyjnej
  figura kreator(int num);

Możliwe jest jednak stworzenie: wskaźnika oraz referencji na typ klasy abstrakcyjnej.

Podsumowując:

- klasy abstrakcyjne tworzymy po to, aby odziedziczyć po nich przez klasy pochodne pewien zestaw właściwości wspólny dla kilku różnych klas,
- klasy abstrakcyjne wykorzystujemy do tworzenia wskaźników oraz referencji które pozwolą nam na wskazywanie na obiekty wszystkich klas pochodnych po klasie abstrakcyjnej.

Treść zadania laboratoryjnego:

1. Stwórz klasę element oraz klasy od niej pochodne: rezystor, cewka, kondensator.
2. Wszystkie klasy powinny posiadać publiczny parametr nazwa oraz właściwości niepubliczne: prąd i napięcie (skorzystaj z dziedziczenia). Ponadto zależnie od danego elementu powinny posiadać parametry charakterystyczne: rezystancja, indukcyjność, pojemność (również niepubliczne)
3. Dla wszystkich klas zaimplementuj następujące metody:
   - Konstruktor inicjujący pola nazwa oraz właściwość charakterystyczną danego elementu
   - Metodę void wymuszenie(int napiecie=0, int prad=0) pobierająca parametry wymuszenia (napięcie, prąd) i wypisującą na ekran informację w jakiej klasie jestem i jakie mam wymuszenie. Metoda ma być abstrakcyjna !

   Przykład dla klasy rezystor:

```
void wymuszenie(int arg_prad=0, int arg_napiecie=0)
{
 prad=arg_prad;
 napiecie= arg_napiecie;
 cout << „Jestem elementem klasy REZYSTOR” << endl;
 cout  << „Mam U =” << napiecie << „oraz I =” << prad; }
```

4. Zaimplementuj funkcję zaprzyjaźnioną *int obwod*, posiadającą jako argumenty dwa wskaźniki typu element, która będzie sprawdzała czy wskazane elementy mogą być ze sobą połączone (porównanie prądów i napięć).
   Funkcja ta zwraca wartość typu int:
   1 – elementy połączone szeregowo (identyczne prądy)
   2 – elementy połączone równolegle (identyczne napięcia)
   3 – elementy maja identyczne prądy oraz identyczne napięcia
5. W programie głównym utwórz 5 różnych elementów (za pomocą tablicy wskaźników) a następnie sprawdź które z podanych elementów mogą być połączone z innymi (sprawdź wszystkie kombinacje w pętlach). Wyświetl wyniki sprawdzenia w konsoli.