Zadanie 1

Sprawdź działanie polimorfizmu na przykładzie tablicy figur.

- Utwórz klasę `Point` przechowującą dane o współrzędnych punktu na płaszczyźnie. Klasa powinna posiadać konstruktor parametryczny i funkcję wyznaczającą odległość pomiędzy dwoma punktami. Definicje wymienionych funkcji powinny pozwolić na wykonanie następującego kodu:

```
Point p1(2,4), p2(2,2);
cout  << p1.Length(p2);
```

- Utwórz klasę bazową `Shape`. Klasa ta powinna posiadać metodę służącą do drukowania nazwy obiektu np.:

```
void PrintName()
{
    cout << ”class Shape\n”;
}
```

- Utwórz klasy `Circle, Rectangle, Triangle` będące obiektami potomnymi klasy `Shape`. Obiekty te powinny być opisane przez wierzchołki (wykorzystaj klasę `Point`) i posiadać metody do obliczania pola figury.
- Utwórz obiekty reprezentujące klasy figur i wyświetl ich powierzchnie.
- Zmodyfikuj klasę `Shape` przez dodanie abstrakcyjnej metody do liczenia pola figury np.:

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

- Sprawdź, czy możesz teraz utworzyć obiekt typu `Shape` lub np. `Circle`.
- Zmodyfikuj metody `Area()` obiektów `Circle, Rectangle, Triangle` tak, aby były one wirtualne oraz dodaj do nich wirtualne metody `PrintName()`.
- Utwórz tablicę wskaźników do obiektów typu `Shape` i zainicjalizuj je obiektami `Circle, Rectangle, Triangle` np.:

```
Shape   *tabp[3];
tabp[0] = new Circle(...);
tabp[1] = new Triangle(...);
tabp[2] = new Rectangle(...);
for ( int I=0; I<3; ++I)
{
    tabp[I]->PrintName();
    cout << tabp[I]->Area() << endl;
}
```

- Sprawdź co się stanie gdy usunie się modyfikator `virtual` przy metodach obiektów `Circle, Rectangle, Triangle`.
- Sprawdź działanie funkcji `dynamic_cast<T*>()` np. do policzenia ile obiektów w danej kolekcji jest typu `Circle` (pamiętaj o włączeniu opcji kompilatora RTTI):

```
if( dynamic_cast<Circle*>( tabp[I] ) )
{
    ++count;
}
```

- Czy wiesz dlaczego w punkcie poprzednim używaliśmy `dynamic_cast` a nie `static_cast`?

Zadanie 2

- Dodaj do klasy `Point` konstruktor kopiujący:

```
Point::Point( const Point& p )
{
            x = p.x;
    y = p.y;
    cout << "Point: Kontruktor kopiujący"<<endl;
}
```

- Dodaj również kolejną wersję funkcji wyznaczającej odległość pomiędzy dwoma punktami. Tym razem jednak obiekt Point powinien być przekazywany do funkcji przez kopiowanie a nie przez referencje:

```
double Point::Length2( const Point p )
{
    return sqrt( (x - p.x) * (x - p.x) + (y - p.y) * (y - p.y ) );
}
```

- Sprawdź czym różnią się wywołania tych funkcji:

```
Point pa(10, 20), pb(30, 40);
cout<<"Length()"<<endl;
cout<<pa.Length(pb)<<endl;
cout<<"Length2()"<<endl;
cout<<pa.Length2(pb)<<endl;
```

- Spróbuj zmienić wartość obiektu `p` wewnątrz funkcji `Length2` np.: `p.x = 100;`. Zmień na chwilę tak definicję tej funkcji aby taka operacja była możliwa. Która wersja funkcji `Length()` jest najbardziej optymalna?

Zadanie 3 (Nie wiem czy jest sens z racji różnych wersji programów)

Poznaj swoje środowisko pracy. Odnajdź i przetestuj działanie następujących ikon paska narzędziowego: - wstawianie i usuwanie tabulatorów. - wstawianie i usuwanie komentarza. - sterowanie zakładkami.

Dostosuj środowisko programistyczne do własnych potrzeb: poprzez menu ***Toolbox*** otwórz okno ***Customize...*** . * Wybierz zakładkę ***Toolbars*** i dodaj lub usuń z menu dowolny pasek narzędzi (np. ***Debug***). * Wybierz zakładkę ***Commands***. Na liście ***Categories*** zaznacz pozycję ***View***, zaś na liście ***Commands*** odszukaj pozycję ***Full Screen*** i przeciągnij ją myszką na dowolny pasek narzędzi w menu edytora. (Spróbuj go na chwilę stamtąd usunąć). * Poprzez przycisk ***Keyboard...*** otwórz okno ***Options***. Na liście ***Show commands containing***: odszukaj skrót klawiaturowy przycisku ***Full Screen***. Sprawdź działanie tego przycisku zarówno poprzez ikonę jak i klawiaturę.

Zadanie 4

Zmodyfikuj program tak aby każda klasa była umieszczona w oddzielnym pliku .h i .cpp.

UWAGA

W przypadku wystąpienia ostrzeżenia: "warning C4541: 'dynamic_cast' used on polymorphic type 'class Shape' with /GR-; unpredictable behavior may result" należy: - wybrać polecenie Project | Properties; - w gałęzi C/C++ wybrać pozycję Language; - włączyć opcję Enable Run-TimeType Info (RTTI);