Lab_9 Funkcje wirtualne.

Klasa Figure jest klasą bazową dla płaskich figur Circle, Triangle, Rectangle. Zawiera *abstrakcyjne* funkcje void area() i void disp(). Funkcja area() służy do obliczania powierzchni figury płaskiej, a disp() – do wyświetlania tego obszaru na monitorze.

```
class Figura
{
public:
	static size_t alloc;  // how many times memory was dynamically allocated for objects
	                      //of derived classes
protected:
	double s;             // Flat figure area
public:
	….....................................
	//class method’s declaration
};
```

Utwórz klasy pochodne Circle, Triangle, Rectangle, które reprezentują figury płaskie: okręg, trójkąt i prostokąt. Dane do każdej figury płaskiej powinny być dostarczane przez sparametryzowane konstruktory. *Przesłoni* metody *wirtualne* area i disp w każdej klasie tak, aby metoda area zliczała pole odpowiedniej figury i przypisywała wynik do zmiennej s, a disp – wyprowadzała nazwę figury. Klasa Rectangle w konstruktorze dynamicznie alokuje pamięć dla zmiennej *dat* i zwiększa o jedynkę zmienną *alloc*. Destruktor powinien zwolnić tę pamięć i zmniejszyć o jedynkę zmienną alloc.

```
class Rectangle : public Figura
{
	double *dat;  //dat[0] - a, dat[1] – b; a, b – side lengths
public:
	// class methods
};
```

Funkcja *main* wykonuje następujący kod:

```
int _tmain(int argc, _TCHAR* argv[])
{
	Figura *ptr = NULL, *ptr_rect = NULL;
	//create a Rectangle object and assign its pointer to the base class pointer
	ptr_rect = new Rectangle (2, 3);

	{
		//create the objects
		Circle cl(2), cl1(3);
		Triangle tr(2, 4);

		srand(time(NULL));

		for(int it=0; it<10; ++it)
		{
			int ind = rand()%4; // now ind is 0, 1, 2, 3 randomly
			switch(ind)
			{
			case 0: ptr = &cl;
				break;
			case 1: ptr = &cl1;
				break;
			case 2: ptr = &tr;
```

```
                        break;
                case 3: ptr = ptr_rect;
                        break;
                };

                // here ptr - base class pointer, randomly points to one of
                // derived class objects. For which objects will the functions area, disp
                //be called ?
                ptr->area();
                ptr->disp();
        }
    }//now cl, cl1 and tr go beyond a scope resolution area, destructors of these objects
     //should be called

    delete ptr_rect;
    ptr_rect = NULL;

    // if we allocate and deallocate memory correctly, alloc should be zero.
    if(Figura::alloc)
        cout << "!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n" << "error: leak of memory\n";

    system("pause");
    return 0;
}


////////////////////////////////////////////////////////////////////////////////////////////
```

Ćwiczenie 2.

Bazowa klasa Rectangle oblicza pole prostokąta S = a*h, gdzie a - długość boku, h - wysokość.

```
class Rectangle
{
protected:
    double S;     // Area
    double a;     // side length
    double h;     // height
public:
    Rectangle(double hh, double aa) : h(hh), a (aa) { S = 0; }
    void Calc() { Disp_T(); Calc_A(); Disp(); }
    void Disp_T() { cout << "Rectangle Area : "; }
    void Calc_A() {     S = a*h; }
    void Disp() { cout << S << endl; }
};
```

Metoda Calc wykorzystuje następujący algorytm:

1. Wyprowadzi na monitor nazwę figury płaskiej - Disp_T ();.
2. Oblicza jej pole – Calc_A();.
3. Wyprowadzi wynik – Disp();

Utwórz klasę pochodną Triangle, która wywołuje metodę Calc() klasy bazowej, przy czym generalnie algorytm obliczenia pola figury pozostaje taki samy, jak i dla klasy bazowej, jednak funkcja Disp_T() klasy pochodnej wyprowadzi "Triangle Area", a Calc_A() oblicza pole trójkąta z podstawą *a* i wysokością *h*. **Proszę używać dynamicznego polimorfizmu. W funkcji main() należy wywołać metody klasy pochodnej za pomocą wskaźnika do klasy bazowej. Nie wolno niczego zmieniać w implementacji metod klasy bazowej, ale można wnosić zmiany w ich deklaracji.**