

Języki Programowania, Laboratorium 8

1. **Zaimplementować klasę Figura**, będącą klasą bazową dla rzeczywistych figur rysowanych w programie. Klasa Figura, przechowuje następujące pola:

- kolor krawędzi figury - użyć typu string,
- grubość krawędzi figury (grubości linii przyjmują wartości rzeczywiste, typ float),

Pola dostępne są tylko dla klas potomnych.

W części publicznej klasa ta zawiera deklarację wirtualnej metody `Wypisz()`.

2. Jako przykład klasy odpowiadającej danemu rodzajowi figur **zaimplementować klasę Okrag**, będącą klasą potomną klasy Figura, zawierającą pola danych potrzebne do odrysowania figury (współrzędne środka, promień) oraz odpowiednie metody. Zdefiniować konstruktor domyślny (wszystkie wartości = 0), jak również konstruktor inicjalizujący wszystkie pola klasy Okrag (wraz z kolorem krawędzi oraz jej grubością).

- Metoda `Wypisz` powinna wypisać na ekran informację "Okrag kolor grubosc (x,y,R)", gdzie kolejno mamy wartości „x y R” w każdej linijce dla osobnego okręgu.

3. **Napisać program, testujący działanie klasy:**

- Stworzyć wskaźnik (wraz z obiektem) `f1` na klasę Figura (używając konstruktora domyślnego). Wypisać (używając metody `Wypisz`).
- Stworzyć obiekt `o1` klasy Okrag (używając konstruktora domyślnego). Wypisać (używając metody `Wypisz`).
- Stworzyć wskaźnik (wraz z obiektem) `o2` na klasę Okrag ($x = 2, y = 2, R = 5$, kolor zielony, grubość 1). Wypisać.

Następnie należy stworzyć nowy wskaźnik na klasę Figura.

```
Figura* wskfig;
```

I przypisać do niego okręgi `o1` i `o2` oraz wywołać na takim wskaźniku metodę `Wypisz`, np.:

```
wskfig = &o1;
```

```
wskfig->Wypisz();
```

Program należy ponownie skompilować i odpalić. Co się dzieje, jeśli usuniemy słowo `virtual` z klasy Figura (należy przetestować i opowiedzieć prowadzącemu)?

Jakie daje nam to możliwości?

- Możemy tworzyć metody przyjmujące referencje, których zachowanie będzie zależne od typu klasy pochodnej.
- Jeśli pojawi się kiedykolwiek konieczność rozszerzenia programu o nowe obiekty, to dodanie nowej klasy będącej pochodną od już istniejącej nie wymaga tysiąca zmian w różnych miejscach programu. Należy dodać klasę Prostokąt, zachowującą się podobnie do poprzedniej, wywołać na niej gotową już metodę `Wypisz`.

Podsumujmy:

Jeśli kompilator natrafi na **obiekt** danej klasy, np. Figura, uruchamia dla niego funkcję składową z właściwej mu klasy.

Jeśli kompilator natrafi na **wskaźnik lub referencję** klasy Figura, ma dwie możliwości wywołania funkcji składowej:

1) Skoro jest to wskaźnik do klasy Figura, wywołuje metodę składową klasy Figura. Jest to domyślne zachowanie kompilatora.

2) Kompilator widzi, że jest to wskaźnik do klasy Figura, ale zamiast na ślepo sięgać do klasy Figura *używa swojej inteligencji*: nie daje się zwieść typem i sprawdza, na co faktycznie wskaźnik wskazuje. Wtedy może się zorientować, że wskaźnik tak naprawdę wskazuje na obiekt klasy pochodnej Okrag, zatem **orientując się według typu obiektu** uruchamia funkcję składową okręgu. Takie zachowanie wymusza słowo kluczowe virtual przed nazwą funkcji składowej.

4 Następnie **zamieniamy metodę wirtualną z klasy podstawowej na czysto wirtualną**

```
virtual void Wypisz() = 0;
```

i poprawiamy program tak, by się skompilował – czego nie możemy wtedy w programie zrobić?

Wskazówka: wszystkie metody czysto wirtualne muszą istnieć w klasach pochodnych.

Klasa, która ma co najmniej jedną funkcję czysto wirtualną nazywamy klasą **abstrakcyjną**.

Po co tworzyć klasy abstrakcyjne?

Czasem mamy jakiś obiekt, który łączy cechy kilku innych (jak np. nasza klasa Figura) ale sama nie przedstawia swoją istotną żadnego konkretnego obiektu. Mamy okręgi, prostokąty, trójkąty i inne – jak by mógł zareagować matematyk, gdybyśmy kazali policzyć pole *figury*?

Klasa abstrakcyjna jest klasą jakby nieskończoną. Jej dokończenie realizowane jest przez klasy pochodne.

Należy zwrócić jeszcze uwagę na pewną zasadę, którą należy stosować:

Jeśli klasa deklaruje jedną ze swoich funkcji jako virtual, wówczas jej destruktor deklarujemy także jako virtual. Skoro w klasie deklarujemy jakąś funkcję wirtualną, to znaczy, że na obiekty klas pochodnych zamierzamy czasem mówić jak na obiekty klasy podstawowej, co przy późniejszym niszczeniu obiektów mogłoby być problemem – nie zwalniałybyśmy pamięci dla niektórych składników klas pochodnych.

5. Należy dopisać do klasy Okrąg metody Zapisz.

- Metoda Zapisz(char*) powinna zapisać do pliku, którego nazwa podana jest jako jej parametr, współrzędne środka okręgu oraz promień.

Zapisywanie do pliku:

```
#include <fstream>
```

```
ofstream ofile;
```

```
ofile.open("file.txt");
```

```
ofile<<"aaa"<<123<<endl;
```

```
ofile.close();
```

- Zapisać dane obiektu o2 do pliku „figury.txt”.
- W funkcji main należy zapisać ostatnio stworzoną figurę do pliku, którego nazwa jest podana jako **pierwszy parametr wywołania programu**. Należy mu również ustawić grubość krawędzi figury podaną jako **drugi parametr** wywołania programu (uwaga, program powinien wypisywać odpowiednie ostrzeżenie, jeśli liczba podanych parametrów jest nieprawidłowa!)

Parametry wywołania programu (zmiany w deklaracji funkcji main):

```
int main(int argc, char **argv){  
}
```

Gdzie:

argc - liczba parametrów, z którymi został wywołany program

**argv - tablica parametrów (każdy jako char*):

argv[0] - nazwa programu

argv[1], argv[2]... - kolejne parametry wywołania programu

Np. wywołanie programu program mogło by mieć formę:

```
./program Ala 2 3
```

Wtedy:

```
argc == 4
```

```
argv[0] == "program"
```

```
argv[1] == "Ala"
```

```
argv[2] == "2"
```

```
argv[3] == "3" // zmiana "3" na 3 (liczbę) – funkcja atoi. np int a = atoi("3"); z biblioteki <stdlib.h>
```

Przyporządkowanie tych parametrów wykonuje się samoistnie w momencie wywołania programu, jedynym wkładem programisty jest zadeklarowanie funkcji main w formie int main(int argc, char **argv)