Lista 8 – Funkcje wirtualne (i trochę Qt)

W katalogu `Lista8` (TEAMS) znajduje się mój program napisany w Qt. Zainstaluj bibliotekę Qt, jeśli jeszcze jej nie masz, wraz z QtCreatorem. Instalacja (konfiguracja) QtCreatora bywa kłopotliwa, w razie problemów spokojnie można się ze mną umówić na konsultacje. QtCreator musi „widzieć" kompilator, debuger i jakąś wersję Qt. Tzw. „kit" („zestaw narzędzi") musi być w pełni skonfigurowany. Otwieramy w nim pliki `*.pro` („projekt").

1. Rozszerz ten program tak, by naciśnięcie jakiejś kombinacji klawiszy, np. `ctrl+Z`, powodowało usunięcie ostatnio dodanego kółka.
   Wskazówka. Przejrzyj dokumentację klasy `QWidget` (np. klawisz F1, gdy kursor znajduje się w nazwie klasy) i wybierz odpowiedni „event" (to musi być funkcja wirtualna), który przesłonisz w klasie pochodnej. Ten „event" musi mieć coś wspólnego z obsługą klawiatury, rzecz jasna.
2. Wybierz dowolny **inny** wirtualny „event" klasy `QWidget`. Przesłoń jego implementację w sposób, który będzie widoczny podczas działania programu. Np. ruch kółka myszy, albo ruch myszy(?), albo dwuklik, zmiana rozmiaru okna itp. może zmieniać wielkość kółek albo ich kolor, etc. *The sky is the limit*, ale nie siedź nad tym ćwiczeniem za długo.

Quiz w stylu testu kwalifikacyjnego

Załóżmy, że mamy klasę bazową `B` (jak *Base*) i wyprowadzoną z niej klasę pochodną `D` (jak *Derived*), obiekt `b` klasy `B` oraz `d` klasy `D`, wskaźniki `pb` i `pd` na obiekty klasy `B` i `D`, oraz wskaźnik void*:

```
class B;
class D: public B;
B b;
D d;
B* pb = &b;
D* pd = &d;
void* pv;
```

1. Które z poniższych instrukcji są prawidłowe:

   ```
   a) pb = pd;
   b) pd = pb;
   c) pv = pb;
   d) pb = pv;
   ```

2. Które z poniższych instrukcji są prawidłowe:

   ```
   a) pb = &b;
   b) pb = &d;
   c) pd = &b;
   d) pd = &d;
   ```

3. Jeżeli funkcja `f` ma sygnaturę `void f(const B & x);` , to prawidłowe jest jej wywołanie:

   ```
   a) f(b);
   b) f(&b);
   c) f(d);
   d) f(*pv);
   ```

4. Jeżeli funkcja `f` ma sygnaturę `void f(D & x);` , to prawidłowe jest jej wywołanie:

   ```
   a) f(b);
   b) f(d);
   c) f(&b);
   d) f(&d);
   ```

5. Jeżeli funkcja `f` ma sygnaturę `void f(B x);` , to prawidłowe jest jej wywołanie:

   ```
   a) f(b);
   b) f(d);
   c) f(*pb);
   d) f(*pd);
   ```

6. Jeżeli funkcja `f` ma sygnaturę `void f(D x);` , to prawidłowe jest jej wywołanie:

```
a) f(b);
b) f(d);
c) f(*pb);
d) f(*pd);
```

7. Załóżmy, że obie klasy `B` i `D` posiadają funkcję składową `g()`, która w klasie bazowej (`B`) jest zadeklarowana jako wirtualna. Wówczas:
   a) Funkcja `D::g()` automatycznie jest również traktowana jako wirtualna
   b) wyrażenie `b.g()` wywoła funkcję `B::g();`
   c) wyrażenie `d.g()` wywoła funkcję `B::g();`
   d) wyrażenie `d.g()` wywoła funkcję `D::g();`
   e) wartość `sizeof(b)` jest większa niż gdyby klasa `B` nie miała żadnej funkcji wirtualnej.
   f) W ramach instrukcji
   ```
   std::vector <B*> v = {&b, &d};
   for (const auto & p: v)
   {
     p->g();
   }
   ```
   dwa razy wywołana zostanie funkcja `B::g();`

8.
   a) Funkcje wirtualne są łączone (ang.: *bind*) statycznie, a zwykłe funkcje – dynamicznie.
   b) Łączenie funkcji wirtualnych odbywa się podczas działania programu (ang. *at runtime*) na podstawie **stanu** (zawartości) obiektu, na którym są wywoływane, a łączenie zwykłych funkcji składowych wykonywane jest najpóźniej podczas uruchamiania programu na podstawie **typu** obiektu, na którym są uruchamiane.
   c) Jeżeli obiekt klasy pochodnej (D) przekażę do funkcji przez wartość jako obiekt klasy bazowej (B), to wywołując na nim funkcję wirtualną, wywołam funkcję z jego prawdziwej klasy (D).
   d) Jeżeli obiekt klasy pochodnej (D) przekażę do funkcji przez referencję do obiektu klasy bazowej (B), to wywołując na nim funkcję wirtualną, wywołam funkcję z jego prawdziwej klasy (D).