Języki programowania, Czwartek 19.12.2013, 8:15-9:45

Tym razem napiszemy coś bardziej praktycznego – klasę do generowania liczb pseudolosowych z trzech różnych rozkładów (jednorodnego, Gaussa oraz Poissona).

Klasa do generowania liczb losowych – 1 p.
Klasa będzie się nazywać `Random`. Jej plik nagłówkowy powinien zawierać:

- pola `double fMin` i `double fMax` – przedział, z którego będą losowane liczby
- konstruktor domyślny – ustawia ziarno z zegara systemowego: `srand(time(NULL))` oraz `fMin` na 0 i `fMax` na 1
- Konstruktor z parametrami ustawiający pola `fMin` i `fMax; fSeed` analogicznie jak w konstruktorze bez parametrów.
- metoda `void SetSeed(int seed)` – ustawia ziarno `srand(seed)`
- metody „set" i „get" ustawiające pola `fMin` i `fMax`
- metoda `double Exec()` - metoda wirtualna, zwraca liczbę pseudolosową z rozkładu jednorodnego z przedziału <fMin;fMax)

Następnie tworzymy klasę `Gauss`, która dziedziczy z klasy `Random`. Powinna ona zawierać:

- pola typu `double fMean`, `double fSigma`, które przechowują parametry rozkładu Gaussa: średnią i odchylenie standardowe.
- konstruktor z parametrami ustawiający wszystkie pola jak
- metody typu „set" ustawiające oba pola i metody „get" zwracające oba pola
- metodę `double Exec()` - zwracającą liczbę pseudolosową z rozkładu Gaussa o zadanej średniej i odchyleniu standardowym (szczegóły implementacji patrz **Uwaga1!**)

Zestaw bibliotek niezbędnych do stworzenia generatora:
`cstdlib, ctime, cmath`
Należy użyć funkcji `rand()` - zwracającej liczbę całkowitą pseudolosową z rozkładu <0;RAND_MAX), gdzie `RAND_MAX` jest wielką liczbą - stałą zdefiniowaną w bibliotece standardowej.

Piszemy program testujący działanie dwóch wyżej wymienionych klas – 2 p.
Tworzymy w funkcji `main` obiekt klasy `Random` (np. `Rand rand1(-2,3);`) z ustawionymi wartościami od -2 do 3. Losujemy kilka liczb pseudolosowych i wypisujemy na ekran.
Tworzymy w funkcji `main` obiekt klasy `Gauss` (np. `Gauss gaus1(100,5);`) z ustawioną średnią na 100 i odchyleniem 5. Losujemy kilka liczb pseudolosowych.
Tworzymy wskaźnik na obiekt klasy `Random` i ustawiamy go na wcześniej stworzony obiekt tej klasy:
```
Random *rand2 = &rand1;
```
i wypisujemy kilka losowych wartości metodą `Exec()`. Następnie ten sam wskaźnik `rand2` ustawiamy na obiekt klasy `Gauss`:
```
rand2 = &gaus1;
```
i analogicznie wypisujemy kilka losowych wartości metodą `Exec()`. Co się stanie jeśli z klasy bazowej usuniemy słowo kluczowe `virtual`?

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.

W domu lub dodatkowo: można dodać klasę `Poisson` (czytaj **Uwaga2!**), zachowującą się podobnie do poprzedniej, wywołać na niej metodę `Exec()`.

Podsumujmy:
Jeśli kompilator natrafi na **obiekt** danej klasy, np. `Random`, uruchamia dla niego funkcję składową z właściwej mu klasy.
Jeśli kompilator natrafi na **wskaźnik lub referencję** klasy `Random`, ma dwie możliwości wywołania funkcji składowej:
1) Skoro jest to wskaźnik do klasy `Random`, wywołuje metodę składową klasy `Random`. Jest to domyślne zachowanie kompilatora.
2) Kompilator widzi, że jest to wskaźnik do klasy `Random`, ale zamiast na ślepo sięgać do klasy `Random` *używa swojej inteligencji*: nie daje się zwieźć typem i sprawdza, na co faktycznie wskaźnik wskazuje. Wtedy może się zorientować, że wskaźnik tak naprawdę wskazuje na obiekt klasy pochodnej `Gauss`, zatem **orientując się według typu obiektu**

uruchamia funkcję składową okręgu. Takie zachowanie wymusza słowo kluczowe virtual przed nazwą funkcji składowej.

Tworzymy klasę abstrakcyjną (ATD – abstrakcyjny typ danych) – 1 p.
Tworzymy klasę `Uniform`, do której „przerzucamy" z klasy `Random` pola `fMin`, `fMax` i algorytm metody `Exec()` losujący liczbę pseudolosową z rozkładu jednorodnego (powinna posiadać konstruktor z parametrami ustawiającymi `fMin` i `fMax`). Klasa dziedziczy z klasy `Random`, w której pozostało tylko pole fSeed**.**
W klasie `Random` tworzymy teraz metodę `Exec()` jako czysto wirtualną, tzn w pliku Random.h:
`virtual double Exec() = 0;`
Poprawiamy następnie program główny w taki sposób, by się kompilował (czego nie można zrobić w programie w takim przypadku?)
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 `Random`) ale sama nie przedstawia swoją istotną żadnego konkretnego obiektu. Mamy różne inne klasy służące do losowania konkretnych liczb w konkretny sposób: jednorodnie, z rozkładu Gaussa, Poissona, i inne.
Klasa abstrakcyjna jest klasą jakby niedokończoną. 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 zwalnialibyśmy pamięci dla niektórych składników klas pochodnych.

Obsługa plików tekstowych – 1 p.
W programie zapisujemy kilka wylosowanych liczb z rozkładu Gaussa do pliku tsktowego. Przykład zapisywania do pliku:

```
#include <fstream> //naglowek zawierajacy funkcje C++ do operacji na plikach

...

ofstream ofile;
ofile.open("file.txt"); //"file.txt" jest tutaj typu char*
ofile<<"aaa"<<123<<endl;
ofile.close();
```

Następnie wczytujemy zapisany plik i wypisujemy wszystkie liczby na ekran. Przykład wczytywania liczb z pliki:

```
#include <fstream>

...

ifstream ifile;
int val;
ifile.open("file.txt");
while(ifile>>val)
{
      cout<<"val: "<<val<<endl;
}
ifile.close();
```

Uwaga 1!
Algorytm do generowania liczb pseudolosowych z rozkładu Gaussa (metoda Box-Muller'a):

- U, V – dane liczby pseudolosowe z rozkładu jednorodnego <0;1)
- Liczba losowa z rozkładu Gausa o średniej μ i odchyleniu standardowym σ dana jest wtedy wzorem:

$$x = \mu + \sigma\sqrt{-2\ln(U)}\cos(2\pi V)$$

Uwaga 2!
Algorytm do generowania liczb pseudolosowych z rozkładu Poissona o średniej λ :

- $x_0, x_1, \ldots$ - ciąg liczb pseudolosowych z rozkładu jednorodnego
- Jeżeli $x_0 x_1 \ldots x_k < e^{-\lambda}$, to k jest liczbą z rozkładu Poissona.