Zadanie 7, Języki Programowania, 20.11.2011

Operator << (za wykładem)

Klasa biblioteczna ostream (ang. output stream) – strumień wyjściowy pracuje na typach wbudowanych (int, double, float, char, char*, ...).

To dlatego możliwe jest użycie operatora << dla typów wbudowanych:

```
cout<<"Temperatura wynosi "<<22.5<<"stopni C"<<endl;
```

Nie jest jednak możliwe napisanie:

```
cout<<jablko1<<endl;
```

Kiedy jablko1 jest obiektem klasy Owoc (można co najwyżej wypisać odpowiednie pola tej klasy o ile są one publiczne..)

Przykład użycia:

```
ostream& operator<<(ostream& wyjscie, Wektor& w) {
  wyjscie<<"<w.x<", "<<w.y<<")";
return wyjscie;
}
```

Operator przypisania = (za wykładem)

Dwuargumentowy operator przypisania = służy do przypisanie jednemu obiektowy klasy treści drugiego obiektu tej samej klasy.

```
klasa & klasa::operator=(klasa &);
```

Jeśli operator= nie zostanie zdefiniowany, to zostanie on automatycznie wygenerowany (**podobnie jak konstruktor kopiujący**!), przepisane zostaną pola "składnik po składniku" (podobnie jak w przypadku automatycznie generowanego konstruktora kopiującego). W rezultacie takiego przypisania będą dwa obiekty o bliźniaczej treści.

Kłopoty natomiast mogą pojawić się wtedy, kiedy **składnikami klasy są wskaźniki** lub jeśli klasa używa **operatora new do rezerwacji miejsca w zapasie pamięci**.

Operator przypisania składa się z części:

(1) sprawdzenie, czy nie jest kopiowane siebie samego

(2) części "destruktorowej" (np. zwalnianie pamięci)

(3) części "konstruktorowej", przypominającej konstruktor kopiujący

Przykład użycia:

Jeśli operator znajduje się wewnątrz klasy:

```
  NowaKlasa & operator=(NowaKlasa & u){;
    if (&u == this) return *this; //sprawdzenie, czy nie kopiuje samego siebie
    delete X; //zwalniamy pamięć dla rzeczy, które mogły mieć poprzednio zaalokowaną pamięć
    //alokujemy nową pamięć i przepisujemy wartości tak samo jak w konstruktorze kopiującym
    return *this;
 }
```

Treść zadania

1. Należy stworzyć klasy `Pracownik` oraz `Baza` (w oddzielnych plikach .h i .cpp, do tego main w oddzielnym pliku – razem 3 pliki). Klasa pracownik powinna zawierać dwa prywatne pola składowe

- `fNazwisko (std::string)`
- `fPensja (int)` oraz konstruktor główny (z dwoma parametrami).

Klasa `Baza` natomiast powinna zawierać tablicę Pracowników, oraz ich ilość:

- `fSpis (Pracownik*)`
- `fLiczbaPracownikow (int)`

Domyślnie posiadamy 0 pracowników. Ponadto klasa zawiera zestaw metod: konstruktor z 1 parametrem (przyjmujący obiekt Pracownik) konstruktor kopiujący oraz metody:

```
void DodajPracownika(std::string, int) //podajemy nazwisko oraz pensje nowego pracownika
double SredniaPensja() //zwracajace srednią pensje pracowników zapisanych w bazie
```

Do tego należy **zaprzyjaźnić klasę** Pracownik z klasą Baza, by móc odwoływać się do prywatnych pól składowych. Należy również przeciążyć dwa operatory: operator przyrównania ("=") oraz operator wypisania na ekran: ("<<") dla klasy Baza.

2. Tworzenie programu

Uwaga: w przypadku obiektów typu „`string`” nie należy używać funkcji typowych dla pracy z ciągami znaków z C (`strcpy`, `strcmp`). Funkcje biblioteki `<cstring>` używamy dla obiektów typu char*.

Program powinien być kompilowany przy pomocy komendy „`make`” (należy stworzyć odpowiedni Makefile!)

1. Należy stworzyć trzy puste pliki: uczen.cpp, uczen.h oraz main.cpp. Do tego należy stworzyć Makefile (kolejny plik tekstowy, o nazwie „**Makefile**”) z treścią:

```
all:
	g++ baza.cpp main.cpp –o program
```

w pliku program.cpp należy dodać funkcję `int main()` zwracającą 0.
Należy skompilować tak przygotowany program używając w linii poleceń komendy: `make`.

2. Od teraz **zawsze** należy owijać definicje klas w plikach nagłówkowych w strukturę „ifndef”:

```
#ifndef _NAZWA_TOKENU
#define _NAZWA_ TOKENU
	class klasa{...}; - definicja naszej klasy
#endif
```

Działa to w ten sposób: jeśli token _NAZWA_TOKENU nie był jeszcze definiowany: wejdź do środka (zdefiniuj token i zapisz go sobie w pamięci, po czym wykonaj wszystkie instrukcje znajdujące się w środku). Jeśli już go deklarowałeś – przejdź od razu do #endif. W ten sposób już nigdy nie będziemy się przejmować kolejnością ładowanych bibliotek – jeśli zostały one wcześniej załadowane, kompilator w ogóle pominie taką instrukcje.

3. Należy w funkcji main po kolei:

- Stworzyć Makefile i owinąć nagłówek funkcji w #ifndef (0.5 p)
- Stworzyć pojedynczy obiekt Pracownik p. Jest to Kowalski, o pensji 3000 zł. Stworzyć pojedynczy obiekt Baza database1 używając konstruktora z jednym parametrem: pracownikiem p. (1 p)
- Należy wypisać bazę na ekran przy użyciu operatora <<. (1 p)
- Należy dodać nowego pracownika (“Nowak”, 1500 zł) do bazy database1 używając metody DodajPracownika. Wypisać ponownie bazę na ekran. (0.5 p)
- Należy stworzyć drugą bazę danych (database2) używając konstruktora kopiującego. (0.5 p)
- Należy dodać pracownika (“Kwiatkowski, 2300 zł) do skopiowanej bazy danych. Należy wypisać nową bazę danych na ekran. (0.5 p)
- Należy przypisać bazie danych database1 wartości składników Bazy database2 używając operatora =. Należy wypisać database1 na ekran. (1 p)
- Dla Bazy database1 należy wypisać średnią pensję pracowników. (0.5 p)