LABORATORIUM PROE, PROJEKT 1

Prowadzący laboratorium: mgr inż. Zbigniew Nasarzewski

KLASA, KONSTRUKTORY I DESTRUKTORY, PRZECIĄŻANIE FUNKCJI I OPERATORÓW

ZADANIE

Zadanie wybrane będzie losowo w trakcie laboratorium z poniższej puli, doprecyzowanie zadania w trakcie rozmowy ze studentem.

Zarządzanie firmą:
1. *Firmowy kalendarz dyżurów** np.: pizzeria, burgerownia, siłownia, …
2. *Zarządzanie zmianą dla pracowników** np.: metra, kolei, autobusów miejskich, …

Organizacja czasu wolnego:
3. *Rodzinny kalendarz treningów:* biegacz, yoga, crossfit, basen itd.
4. *Domowy planer rozrywek:* konsola, telefony, TV, komputer itd.

Baza danych:
5. *Katalog** np.: gier, książek, …
6. *Sklep/serwis**np.: rowerowy, samochodowy, narciarski, sprzęt komputerowy, zoologiczny, …
7. *Książka adresowa*

*-zadeklaruj swój rodzaj ‘przestrzeni’ zdarzeń – podkreśl lub podaj inny rodzaj – potwierdź z prowadzącym

Do każdego z nich należy przygotować interfejs umożliwiający przetestowanie zaimplementowanych możliwości. W projekcie proszę zwrócić szczególną uwagę na aspekt praktyczny aplikacji, jej użyteczność i sensowność zaprojektowanych elementów.

INFORMACJE SZCZEGÓŁOWE

Aplikacja ma być oparta na zestawie klas, z których głównym obiektem będzie element, którym zarządzamy. Obiekt ten ma być złożony z minimum 3 podobiektów, w tym co najmniej jednego tworzonego **dynamicznie** i jednego tworzonego **automatycznie**. Odwzorowanie powinno być możliwie realistyczne - dla skomplikowanych obiektów odpowiednio uproszczone.

We wszystkich konstruktorach i destruktorach należy wstawić kod drukujący na ekran informację o ich wywołaniu. Wyświetlenie to ma być warunkowe – jedynie w momencie zdefiniowania

zmiennej kompilacji **_DEBUG**. Wydruki te będą pomocne w czasie śledzenia sekwencji wywołania konstruktorów i destruktorów.

Klasa główna ma zawierać mechanizm określania liczby stworzonych obiektów tego typu (**statyczne pole** klasy), oraz **statyczną metodę** zwracającą to statyczne pole klasy.

Każda klasa powinna prawidłowo zachowywać się w przypadku **kopiowania.** Należy rozważyć realizację konstruktora kopiującego lub użycie standardowego konstruktora kopiującego. Podobnie rozważyć operator **przypisania** dla klas.

Proszę zaprojektować i zaimplementować dla klas kilka **sensownych, różnorodnych** operatorów (minimum 8), w tym: jednoargumentowe, dwuargumentowe, konwersji, przypisania, indeksowe. Należy zastosować wybrane operatory jako metody klas oraz jako funkcje zaprzyjaźnione z klasami – ale tylko tam gdzie jest to niezbędne. Zastanów się w jaki sposób można **zablokować** możliwość wywołania dowolnego operatora (np. a+b).

Napisać program główny testujący klasę główną i jej podklasy (oddzielny moduł/plik). Dla testów należy stworzyć obiekty **automatyczne, dynamiczne i statyczne (lokalne, globalne)** w **funkcji testowej** wywoływanej z funkcji main. Celem powyższych testów jest między innymi obserwowanie **czasu życia obiektów** oraz zachowanie się funkcji i operatorów przeciążonych.

Każdy z programów powinien umożliwiać odczyt i zapis do pliku tekstowego (jednego lub kilku) zawartości obiektów rozpatrywanych w danym projekcie – powinno być możliwe odtworzenie stanu aplikacji po jej ponownym uruchomieniu, konieczna implementacja stosownych metod zapisu i odczytu danych.

W osobnej funkcji, wywoływanej w funkcji main jedynie przy ustawionej zmiennej kompilacji _DEBUG, należy przetestować wszystkie zaimplementowane operatory.

Na każdą klasę powinny przypadać 2 pliki - plik nagłówkowy .h i plik definicji .cpp.

UWAGA

Jeżeli jest wybór pomiędzy stosowaniem mechanizmów, funkcji, instrukcji typowych dla języka C i C++ proszę stosować odpowiednie konstrukcje właściwe dla C++ np. char* - string, FILE* - iostream, itp. Jeden obiekt – 2 pliki: obiekt.h, obiekt.cpp.

Proszę przesyłać projekty 2 dni przed terminem obrony (tj. niedziela do godz. 24 w tygodniu obrony dla grupy wtorkowej) na adres mailowy prowadzącego zajęcia: **nasarzewski@ire.pw.edu.pl**.

KRYTERIA OCENY

czytelność kodu i brak wycieków pamięci	3 p.
sensowność konstrukcji klas i użyteczność aplikacji	6 p.
spełnienie pozostałych założeń	6 p.