LABORATORIUM PROE.B, PROJEKT 1

KLASA, KONSTRUKTORY I DESTRUKTORY, PRZECIĄŻANIE FUNKCJI I OPERATORÓW

ZADANIE

Zaprojektować aplikację zarządzania gdańska elektrownia

Aplikacja będzie rozwijana w trakcie trzech kolejnych projektów. Przed przystąpieniem do realizacji projektu należy samodzielnie zdecydować w jakiej większej aplikacji obiekt zostanie wykorzystany, np. gra, program biurowy, program zarządzający, symulator itp. Będzie to warunkowało architekturę implementowanych obiektów.

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**. Obiekt główny powinien umożliwiać tworzenie i usuwanie podobiektów dynamicznych, a wszystkie metody z nich korzystające muszą sprawdzać ich istnienie (ewentualnie modyfikując swoje działanie). 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 wykorzystać do śledzenia i zapoznania się z sekwencją wywołania konstruktorów i destruktorów oraz czasem życia obiektów.

Klasa główna powinna zawierać mechanizm określania **liczby stworzonych** oraz **liczby aktualnie istniejących** obiektów tego typu: **statyczne pola** klasy oraz **statyczne metody** zwracające (nie drukujące na ekran!) wartość odpowiedniego pola.

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

Należy zaprojektować i zaimplementować dla klas kilka **sensownych, różnorodnych** operatorów (minimum 10), w tym: jednoargumentowe, dwuargumentowe, konwersji, przypisania, indeksowe. Należy zastosować wybrane operatory jako metody klas lub jako funkcje zaprzyjaźnione z klasami – ale tylko tam gdzie jest to niezbędne.

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** (należy do tego wykorzystać wydruki z konstruktorów/destruktorów jak i liczniki obiektów), **poprawności kopiowania** (konstruktorem kopiującym i operatorem przypisania) oraz zachowanie się funkcji i operatorów przeciążonych. Program testujący powinien być aplikacją bezobsługową, tzn. nie powinien wymagać interakcji z użytkownikiem (np. wprowadzania danych z konsoli).

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++ należy stosować odpowiednie konstrukcje właściwe dla C++ np. char* – string, FILE* – iostream, itp.

Wszystkie projekty powinny być umieszczone w serwisie GitLab (https://gitlab-stud.elka.pw.edu.pl) co najmniej 2 dni przed terminem obrony (tj. poniedziałek do godz. 24 w tygodniu obrony dla grupy środowej). Należy pamiętać o dołączeniu do projektu (Settings -> Members) prowadzącego zajęcia, co najmniej na poziomie Reporter.

KRYTERIA OCENY

czytelność kodu	1 p.
sensowność konstrukcji klas	3 p.
brak wycieków pamięci	1 p.
spełnienie pozostałych założeń	5 p.