My Project

Generated by Doxygen 1.9.3

Chapter 1

Laboratorium 6 - Szablony klas i szablony funkcji

1.0.1 Tresc zadan dla Panstwa:

Zadanie 0: absolutnie obowiazkowe, chociaz bez punktow

1. Pierwsza rzecza jest poprawa bledow kompilacji, czyli wpisanie poprawnych Panstwa danych w pliku: main.↩ cpp
2. Oddanne zadanie musi sie bezwzglednie kompilowac na systemie Linux:
   - Jesli sie nie skompiluje to jest to 0 punktow za zadanie!
   - Oczywiscie w razie problemow z kompilacja prosze się zgłaszać/pisać.
   - Dobrze, jesli nie byloby warningow kompilacji, ale za to nie obnizam punktow.
   - Aby się upewnić, że się kompiluje można skorzystać z `narzędzia online judge` (VPN AGH konieczny). Aby wyslac zadanie nalezy wybrac odpowiednie dla zajec: konkurs (`context`), problem, oraz jezyk programowania. prosze zalaczyc pliki:
     - myList.h i mySorting.cpp
     - proszę nie załączać: main.cpp
3. Oddane zadanie nie powinno crashować na żadnym teście, jeśli crashuje proszę zrobic implementacje -fake, ktora nie dopusci do crasha nawet jesli test bedzie failowal, ale za to testy nie beda sie crashowaly. W przypadku crasha biorę pod uwagę tylko tyle testów, ile przejdzie do czasu crasha!
4. Mam program antyplagiatowy, dlatego prosze pracowac samodzielnie!
   - Osoby ktore udostepniaja swoje rozwiazania rowniez beda mialy kare!
   - Na ukaranie prowadzący ma czas 2 tygodnie po terminie oddania, czyli nawet jak ktoś otrzyma punkty wcześniej ma pewność, że za oszustwa/łatwowierność dosięgnie go niewidzialna ręka sprawiedliwości.
5. Zadanie z zalozenia bedzie sprawdzane automatycznie, przez testy jednostkowe dostepne w pliku: shapes↩ Tests.cpp,
6. ∗Dobrze jakby nie bylo warningow kompilacji (flagi: `-Wall -Wextra -pedantic -Werror`, a dla hardcorów jeszcze: `-Weffc++`)
7. Punkty mogą być odejmowane za wycieki pamieci (jest podpiety `valgrind`)
8. Niewykluczone jest sprawdzanie reczne - zaleznie od prowadzacego dana grupe.

1.0.2 Zadanie implementacyjne:

1. Prosze utworzyć plik mylist.h, oraz dokonac nastepujacej implementacji: Proszę o zaimplementowanie szablonu klasy `MyList<T>`, reprezentujacej liste jednokierunkowa z glowa i iteratorami. Punktacja przydzielana za nastepujace metody (jak testy):
   (a) za konstruktor bezargumentowy i metode `size()` zwracajaca ilosc elementow
   (b) za metody `push_front` i `pop_front`, ktore odpowiednio dodaja/usuwaja element z poczatku
   (c) metode `front()` zwracajaca element na poczatku, oraz aby `pop_front()` zwracala usuniety element.
   Note
   Standardowo w `std::list` metoda `pop_front()` nic nie zwraca. Jak myślisz - dlaczego?
   (d) jesli pierwszy wezel (o nazwie head), oraz kazdy nastepny wezel (head->next) sa zaimplementowane przy pomocy `std::unique_ptr<MyList::Node>`
   (e) jesli w razie zawolania `pop_front` na pustej liscie zostaje wyrzucony wyjatek `std::out_of_range`
   (f) jesli kopiowanie (konstruktor kopiujacy i operator przypisania) jest niemozliwe dla listy
   (g) jesli mamy zaimplementowane metody iteratora (tutaj jeszcze nie musza w pelni dzialac, chociaz powinny zwracac co nalezy)
   (h) jesli napisany iterator dziala z petla for-zakresowym
   (i) jesli nasz iterator dziala z algorytmami standardowymi na przykladzie `std::count_if`
   Note
   Do tego wymagane jest kilka aliasów typów.

jesli mamy metode `remove(T element)`, ktora usuwa wszystkie elementy z listy o danej wartosci

1. jesli lista ma operator wypisywania na strumien (forma wydruku dowolna, byleby byly wszystkie elementy)

Prosze o utworzenie pliku mySorting.h, W nim prosze o zaimplementowanie szablonu funkcji globalnej `void mySort(???)`: Punktacja (analogicznie jak testy):

1. Sortowanie statycznej tablicy dziala
2. Dziala z kontenerami standardowymi (na przykladzie `std::vector`)
3. Dziala z nasza lista - specjalizacja
4. Specjalizacja sortowania dla tablicy `char[][]` jesli dziala dla tablicy slów skladajacych sie wylacznie z DUZYCH LITER
5. Jw. ale powinno dzialac z pominieciem wielkosci liter.

Tym razem kod ma się kompilować z flagami: `-Wall -Wextra -pedantic -Werror` dla hardcorów jeszcze: `-Weffc++`

1.0.3 Uwaga:

1. Konieczne może się okazać zrobienie dwóch wersji metod begin/end -jedna stała, druga nie.
2. Należy zdefiniować dwie wersje iteratorów - stały `const_iterator` i zwykły `iterator` jako klasy zagnieżdżone.
   (a) Informacje `jak zdefiniować własny iterator` lub `2`. Najprościej jest dziedziczyć po `std::iterator`, niemniej jednak jest to deprecated.
3. Szablony muszą być zdefiniowane w całości w pliku nagłówkowym, jednakże proszę aby definicje metod dłuższych niż 1-linijkowe były pod klasą.
4. Można użyć `std::sort` lub `std::stable_sort` - tylko trzeba wiedzieć gdzie i jak.