My Project

Generated by Doxygen 1.9.3

1 Laboratorium 6 - własny typ tekstowy MyString **1**

1.0.1 Tresc zadan dla Panstwa: 1

1.0.2 Zadanie implementacyjne: 2

1.0.3 Uwaga: 3

Chapter 1

Laboratorium 6 - własny typ tekstowy MyString

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:
     - mystring.h i mystring.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. Wykorzystanie biblioteki STL i kontenerów podczas implementacji klasy MyString (ile testów przejdzie, tyle punktów)
2. Klasa ta ma zawierać statyczna tablice na tekst do 20 znaków, a resztę ma w razie potrzeby pobierać dynamicznie np. przy pomocy typu `std::string`.
3. Funkcjonalności są dopasowane tak, aby użyć kilku kontenerów standardowych, poćwiczyć pisanie iteratorów, oraz użyć algorytmów uogólnionych (`<algorithm>`).
4. Tresc nalezy wydedukowac w oparciu o plik z testami, nad wieloma testami jest sugestia czego można użyć.
5. Można też użyć biblioteki boost (oczywiście wtedy w razie błędów kompilacji na sprawdzarce proszę o maila z informacją czego Państwo używają).
6. Sygnatury funkcji do zaimplementowania (pobrane przy pomocy `ctags`):

metoda	sygnatura
MyString	MyString(const char *text)
MyString	MyString(const MyString &text)
begin	iterator begin()
begin	const_iterator begin() const
capacity	auto capacity() const
cbegin	const_iterator cbegin() const
cend	const_iterator cend() const
const_iterator	explicit const_iterator(const MyString* myString, size_t position)
empty	bool empty() const
end	iterator end()
end	const_iterator end() const
getPosition	auto getPosition() const
iterator	explicit iterator(MyString* myString, size_t position)
operator !=	bool operator!=(const MyString& rhs) const
operator !=	bool operator!=(iterator anotherIt)
operator !=	bool operator!=(const_iterator anotherIt) const
operator *	char& operator*()
operator *	char operator*() const
operator +	iterator operator+(size_t pos)
operator +	const_iterator operator+(size_t pos) const
operator ++	iterator& operator++()
operator ++	const_iterator& operator++()
operator -	size_t operator-(iterator anotherIt)
operator -	size_t operator-(const_iterator anotherIt) const
operator –	iterator& operator--()
operator –	const_iterator& operator--()
operator ==	bool operator==(iterator anotherIt)
operator ==	bool operator==(const_iterator anotherIt) const
operator []	char operator[ ](size_t i) const
operator std::string	explicit operator std::string() const
push_back	void push_back(char c)
size	auto size() const
map<MyString, size_t>	countWordsUsageIgnoringCases() const;
all_of	bool all_of(std::function<bool(char)> predicate) const

metoda	sygnatura
generateRandomWord	static MyString generateRandomWord(size_t length)
getUniqueWords	std::set<MyString> getUniqueWords() const
join	MyString join(const std::vector<MyString> &texts) const
startsWith	bool MyString::startsWith(const char *text) const
toLower	MyString& toLower()
trim	MyString& trim()

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

1.0.3 Uwaga:

1. Konieczne jest zrobienie dwóch wersji metod begin/end -jedna stała, druga nie. Podobnie dwóch wersji iteratora.
2. Informacje `jak zdefiniować własny iterator` lub `2`.