Laboratoria Programowania w C++

Generated by Doxygen 1.9.1

Chapter 1

Laboratorium 6 - Szablony klas i szablony funkcji

1.0.1 Tresc zadan dla Panstwa (aktualniejsza jest w <tt>README.md</tt>)

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 Bobot`
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↩ : `shapesTests.cpp`,
6. ∗Dobrze jakby nie bylo warningow kompilacji (flagi: `-Wall -Wextra -pedantic -Werror`, a dla hardcorów jeszcze: `-Weffc++`)
7. Punkty beda odejmowane za wycieki pamieci (jest podpiety `valgrind`)
8. Niewykluczone jest sprawdzanie reczne - zaleznie od prowadzacego dana grupe.

1.0.2 Zadanie implementacyjne:

W zadaniu chodzi o to, aby Państwo zaimplementowali szablon klasy, wraz z odpowiednimi metodami. Następnie aby Państwo zaimplementowali szablon funkcji.

1. Prosze utworzyć plik `myList.h` (ważna jest wielkość liter), 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 (usunięte) 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, szczegóły w internecie lub na wykładzie.

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)

Następnie prosze o utworzenie pliku `mySorting.h` (wielkość liter ma znaczenie). 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 słów 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` a dla hardcorów jeszcze: `-Weffc++`

Aby odblokować odpowiednie testy należy po dokonaniu implementacji wstawić odpowiednie makra w pliku `myList.h`:

```
#define IMPLEMENTED_constructorOfEmptyList
#define IMPLEMENTED_pushingAndPopingElementsFront
#define IMPLEMENTED_nodesStoredAsUniquePtrs
#define IMPLEMENTED_popFromWhenEmptyList
#define IMPLEMENTED_copyingDisabled
#define IMPLEMENTED_removingElements
#define IMPLEMENTED_iteratorOperations
#define IMPLEMENTED_iteratorWithRangedForLoop
#define IMPLEMENTED_iteratorWithStlAlgorithm
#define IMPLEMENTED_ostreamOperator
```

1.0.3 Najczęstrze problemy/wątpliwości/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.
5. Można spróbować użyć `if constexpr` aby zmniejszyć ilość funkcji.
6. Dodałem pliki, ale testy nadal nie przechodzą - trzeba ponownie uruchomić CMake aby wykrył zmiany plików.

Chapter 2

MyList<T> and mySort<T>

W zadaniu chodzi o to, aby Państwo zaimplementowali szablon klasy, wraz z odpowiednimi metodami. Następnie aby Państwo zaimplementowali szablon funkcji.

1. Prosze utworzyć plik `myList.h` (ważna jest wielkość liter), 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.
      - **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 (usunięte) 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`
      - Do tego wymagane jest kilka aliasów typów, szczegóły w internecie lub na wykładzie.

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)

Następnie prosze o utworzenie pliku `mySorting.h` (wielkość liter ma znaczenie). 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 słów 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` a dla hardcorów jeszcze: `-Weffc++`

Aby odblokować odpowiednie testy należy po dokonaniu implementacji wstawić odpowiednie makra w pliku `myList.h`:

```
#define IMPLEMENTED_constructorOfEmptyList
#define IMPLEMENTED_pushingAndPopingElementsFront
#define IMPLEMENTED_nodesStoredAsUniquePtrs
#define IMPLEMENTED_popFromWhenEmptyList
#define IMPLEMENTED_copyingDisabled
#define IMPLEMENTED_removingElements
#define IMPLEMENTED_iteratorOperations
#define IMPLEMENTED_iteratorWithRangedForLoop
#define IMPLEMENTED_iteratorWithStlAlgorithm
#define IMPLEMENTED_ostreamOperator
```

2.1 ## Najczęstrze problemy/wątpliwości/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.
5. Można spróbować użyć `if constexpr` aby zmniejszyć ilość funkcji.
6. Dodałem pliki, ale testy nadal nie przechodzą - trzeba ponownie uruchomić CMake aby wykrył zmiany plików.

`Bardziej szczegółowe informacje jak pisać programy w ładnym stylu` dla zaawansowanych.
Informacje o co chodzi w paczce, na co zwrócić uwagę, jak czytać testy znajdują się w materiale `wideo`. W opisie filmu jest też częściowy minutowy spis treści.

2.2 Podpowiedzi:

1. (Jak macie pomysł to podrzućcie)

2.3 Ocenianie:

1. Ocenia `Bobot`, na ten moment w następujący sposób:
   (a) Kompilacja nadesłanego rozwiązania - bez tego zero punktów. Bobot pracuje na Linuxie, używa kompilatora g++.
   (b) Uruchamianie testów - za każdy test, który przejdzie są punkty, ale mogą być odjęte w kolejnych krokach.
   (c) Jeśli program się wywala na którymś z testów (to się pojawia często u osób pracujących na Windowsie - ten system pozwala pisać po nie-swojej pamięci, Linux nie pozwala) lub jest timeout - wtedy będzie przyznane tyle punktów ile przechodzi testów **minus dwa za karę**.
   (d) Jest odpalane narzędzie `valgrind`, które sprawdza czy umiemy obsługiwać pamięć w praktyce - jeśli nie to **minus punkt**.
   (e) Odpalane są też inne narzędzia takie jak `cppcheck`, czy `fawfinde` i inne. One nie odejmują punktów, no ale mają pomóc w pisaniu porządnych programów. Nie olewajmy tego.
   (f) Antyplagiat - za wykrycie plagiatu (jest specjalne narzędzie) otrzymuje się 0 punktów. Róbmy więc samemu!

2.4 Pytania po implementacji ćwiczenia:

1. (Jak macie pomysł to podrzućcie)

2.5 Zadania, które warto zrobić (uwaga: nie będzie za to punktów, tylko coś cenniejszego - umiejętności)

1. (Jak macie pomysł to podrzućcie)

2.6 Jak skonfigurować sobie pracę nad paczką:

W formie `wideo` do poprzedniej paczki (link do projektu inny, reszta analogiczna).
Alternatywnie poniżej jest to spisane w kolejnej sekcji

2.6.1 Grading (section copied from Mateusz Ślażyński, of course he agreed):

- [ ] Make sure, you have a **private** group
  - `how to create a group`
- [ ] Fork this project into your private group
  - `how to create a fork`
- [ ] Add @bobot-is-a-bot as the new project's member (role: **maintainer**)
  - `how to add an user`

2.6.2 How To Submit Solutions

1. [ ] Clone repository: `git clone` (clone only once the same repository):
   ```bash git clone <repository url> ```
2. [ ] Solve the exercises
3. [ ] Commit your changes
   ```bash git add <path to the changed files> git commit -m <commit message> ```
4. [ ] Push changes to the gitlab main branch
   ```bash git push -u origin main ```

The rest will be taken care of automatically. You can check the `GRADE.md` file for your grade / test results. Be aware that it may take some time (up to one hour) till this file. Details can be found in `./logs/` directory where You can check compilation results, tests logs etc.

2.6.3 Project Structure

```
.
CMakeLists.txt          # CMake configuration file - the file is to open out project in our IDE
main.cpp                # main file - here we can test out solution manually, but it is not required
trescPdf.pdf            # documentation in PDF (generated by Doxygen)
tests                   # here are tests for exercise, inner CMakeLists.txt, GTest library used by tests
   CMakeLists.txt      # iner CMake for tests - it is included by outter CMake
   lib                 # directory containing GTest library
   MyListTests.cpp     # tests for template class
   MySortTests.cpp     # tests for template function
doxyfiles               # here is logo for documentation generated by Doxygen
   cppLogo.png         # logo
Doxyfile                # here is prepared file for Doxygen, to generate documentation when we type `doxygen
Dockerfile              # this file contains instructions how to run tests in embedded Ubuntu
README.md               # this file
```
```

Chapter 3

Todo List

Member FIRSTNAME

Uzupelnij swoje dane:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Class Documentation

7.1 MyListTester Class Reference

Inheritance diagram for MyListTester:
Collaboration diagram for MyListTester:

Public Member Functions

- void setUp ()

7.1.1 Detailed Description

Definition at line 23 of file MyListTests.cpp.

7.1.2 Member Function Documentation

7.1.2.1 setUp()

```
void MyListTester::setUp ( )  [inline]
```

Definition at line 26 of file MyListTests.cpp.
The documentation for this class was generated from the following file:

- tests/MyListTests.cpp

7.2 MySortTester Class Reference

Inheritance diagram for MySortTester:
Collaboration diagram for MySortTester:

Public Member Functions

- template<typename T , size_t N>
  auto returnSortedArray (T(&arr)[N])
- template<typename Container >
  auto returnSortedContainer (const Container &container)

7.2.1 Detailed Description

Definition at line 26 of file MySortTests.cpp.

7.2.2 Member Function Documentation

7.2.2.1 returnSortedArray()

```
template<typename T , size_t N>
auto MySortTester::returnSortedArray (
             T(&) arr[N] )  [inline]
```

Definition at line 30 of file MySortTests.cpp.

7.2.2.2 returnSortedContainer()

```
template<typename Container >
auto MySortTester::returnSortedContainer (
             const Container & container )  [inline]
```

Definition at line 39 of file MySortTests.cpp.

The documentation for this class was generated from the following file:

- tests/MySortTests.cpp

Chapter 8

File Documentation

8.1 CMakeLists.txt File Reference

8.2 tests/CMakeLists.txt File Reference

Functions

- project (tests) add_subdirectory(lib) include_directories($

8.2.1 Function Documentation

8.2.1.1 project()

```
project (
            tests  )
```

Definition at line 1 of file CMakeLists.txt.

8.3 main.cpp File Reference

```
#include <iostream>
```

Include dependency graph for main.cpp:

Functions

- void validateStudentsInfo ()
- int main ()
- constexpr size_t compileTimeStrlen (const char *text) noexcept
- constexpr size_t compileTimeCountFirstDigits (const char *text) noexcept
- constexpr bool compileTimeIsDigit (const char *text) noexcept
- constexpr bool compileTimeContains (const char *text, char letter) noexcept

Variables

- constexpr const char *const FIRSTNAME = "Grzegorz"
- constexpr const char *const SURNAME = "Bazior"
- constexpr const char *const MAIL = "bazior@agh.edu.pl"
- constexpr const char *const BOOK_ID = "123456"
- constexpr const char *const TEACHER_MAIL = "bazior[at]agh.edu.pl"

8.3.1 Function Documentation

8.3.1.1 compileTimeContains()

```
constexpr bool compileTimeContains (
            const char * text,
            char letter )  [inline], [constexpr], [noexcept]
```

Definition at line 133 of file main.cpp.
Here is the caller graph for this function:

8.3.1.2 compileTimeCountFirstDigits()

```
constexpr size_t compileTimeCountFirstDigits (
            const char * text )  [inline], [constexpr], [noexcept]
```

Definition at line 123 of file main.cpp.
Here is the caller graph for this function:

8.3.1.3 compileTimeIsDigit()

```
constexpr bool compileTimeIsDigit (
            const char * text )  [inline], [constexpr], [noexcept]
```

Definition at line 128 of file main.cpp.
Here is the call graph for this function: Here is the caller graph for this function:

8.3.1.4 compileTimeStrlen()

```
constexpr size_t compileTimeStrlen (
            const char * text )  [inline], [constexpr], [noexcept]
```

Definition at line 118 of file main.cpp.
Here is the caller graph for this function:

8.3.1.5 main()

```
int main ( )
```

Definition at line 109 of file main.cpp.
Here is the call graph for this function:

8.3.1.6 validateStudentsInfo()

```
void validateStudentsInfo ( )
```

Definition at line 141 of file main.cpp.
Here is the call graph for this function: Here is the caller graph for this function:

8.3.2 Variable Documentation

8.3.2.1 BOOK_ID

```
constexpr const char* const BOOK_ID = "123456"  [constexpr]
```

Definition at line 103 of file main.cpp.

8.3.2.2 FIRSTNAME

```
constexpr const char* const FIRSTNAME = "Grzegorz"  [constexpr]
```

Todo Uzupelnij swoje dane:

Definition at line 100 of file main.cpp.

8.3.2.3 MAIL

```
constexpr const char* const MAIL = "bazior@agh.edu.pl"  [constexpr]
```
Definition at line 102 of file main.cpp.

8.3.2.4 SURNAME

```
constexpr const char* const SURNAME = "Bazior"  [constexpr]
```
Definition at line 101 of file main.cpp.

8.3.2.5 TEACHER_MAIL

```
constexpr const char* const TEACHER_MAIL = "bazior[at]agh.edu.pl"  [constexpr]
```
Definition at line 104 of file main.cpp.

8.4 README.md File Reference

8.5 tests/MyListTests.cpp File Reference

```
#include <vector>
#include <algorithm>
#include <type_traits>
#include <functional>
#include <cctype>
#include <gtest/gtest.h>
```
Include dependency graph for MyListTests.cpp:

8.6 tests/MySortTests.cpp File Reference

```
#include <iostream>
#include <array>
#include <vector>
#include <algorithm>
#include <gtest/gtest.h>
```
Include dependency graph for MySortTests.cpp:

Classes

- class MySortTester

Macros

- #define MY_SORTING_INCLUDED 0

Functions

- TEST_F (MySortTester, sortingStaticArray_expectedArraySorted)
- TEST_F (MySortTester, sortingStdVector_expectedContainerSorted)
- TEST_F (MySortTester, sortingMyList_expectedElementsInTheListSorted)
- TEST_F (MySortTester, sortingConstCharPtrAllUpperCases_expectedAllWordsSorted)
- TEST_F (MySortTester, sortingConstCharPtrIgnoreCases_expectedAllWordsSortedDespiteDifferentCases)

8.6.1 Macro Definition Documentation

8.6.1.1 MY_SORTING_INCLUDED

```
#define MY_SORTING_INCLUDED 0
```

Definition at line 17 of file MySortTests.cpp.

8.6.2 Function Documentation

8.6.2.1 TEST_F() [1/5]

```
TEST_F (
            MySortTester ,
            sortingConstCharPtrAllUpperCases_expectedAllWordsSorted  )
```

Definition at line 102 of file MySortTests.cpp.

8.6.2.2 TEST_F() [2/5]

```
TEST_F (
            MySortTester ,
            sortingConstCharPtrIgnoreCases_expectedAllWordsSortedDespiteDifferentCases  )
```

Definition at line 119 of file MySortTests.cpp.

8.6.2.3 TEST_F() [3/5]

```
TEST_F (
            MySortTester ,
            sortingMyList_expectedElementsInTheListSorted  )
```

Definition at line 81 of file MySortTests.cpp.

8.6.2.4 TEST_F() [4/5]

```
TEST_F (
            MySortTester ,
            sortingStaticArray_expectedArraySorted  )
```

Definition at line 47 of file MySortTests.cpp.

8.6.2.5 TEST_F() [5/5]

```
TEST_F (
            MySortTester ,
            sortingStdVector_expectedContainerSorted  )
```

Definition at line 64 of file MySortTests.cpp.