ft_containers

C++コンテナ、イージーモード

Summary:C++ で利用可能な複数のコンテナは、すべて非常に異なる使用方法を持っています。それらを理解するために、再実装してみましょう。

バージョン4

内容

第一章 総則

- ヘッダーで実装された関数（テンプレートの場合を除く）、および保護されていないヘッダーは、運動に対して0を意味します。
- クラス名、関数名、メソッド名と同様に、課されたファイル名を忠実に守らなければならない。
- 覚えておいてください：あなたは今C++でコーディングしているのであって、もうCではありません。ですから
    - 以下の関数は禁じ手であり、使用した場合は問答無用で0点です。*alloc、*printf、free。
    - 標準ライブラリにあるものはすべて使ってよいことになっています。しかし、C言語で使い慣れた関数をそのまま使うのではなく、C++的なバージョンを使ってみるのも賢い方法です。
    - 今回の目的はSTLライブラリの再コード化なので、当然ながらコンテナそのものを使うことはできません。
- 実は、明確に禁止されている機能やメカニックの使用は、問答無用で0点という罰則があるのです。
- また、特に断らない限り、C++のキーワードである「using namespace」と "friend "は禁止されています。使用した場合、問答無用で-42の罰則が科せられます。
- クラスに関連するファイルは、特に指定がない限り、常に ClassName.hpp と ClassName.cpp になります。
- 例題は必ず熟読してください。例題には、演習の説明ではわからないような要求が含まれていることがあります。もし、何かが曖昧に思えるなら、あなたはC++を十分に理解していないのです。
- 最初から習ったC++のツールを使うことが許されているので、外部ライブラリの使用は一切禁止されています。また、聞く前に言っておくと、C++11やその派生版、Boostもダメということです。
- 各プロジェクトを始める前に、十分に読んでください。本当に、そうしてください。
- 使用するコンパイラはclang++です。

- あなたのコードは、次のフラグを付けてコンパイルする必要があります。-Wall -Wextra -Werror
  -std=c++98.
- 各インクルードは、他のものと独立してインクルードできる必要があります。インクルードは、当然、依存している他のすべてのインクルードを含んでいなければなりません。
- 因みに、C++では途中からコーディングスタイルが強制されることはない。好きなスタイルで、何の制約もありません。しかし、同僚評価者が読めないコードは、その評価者が採点できないコードであることを覚えておいてください。
- 重要なことプログラムによって採点されることはありません。しかし、怠けてはいけません！彼らが提供する多くのものを見逃してしまいます。
- 提出するものの中に余計なファイルがあっても問題ありませんし、求められている以上のファイルでコードを分けることもできます。その結果がプログラムによって採点されない限り、ご自由にどうぞ。
- たとえ短い課題であっても、自分に何が求められているかを理解し、最善の方法でそれを実行したことを絶対に確認するために、時間をかける価値があります。

• オーディンによって、トールによって!頭を使え!!!

第二章 目的

このプロジェクトでは、C++標準テンプレートライブラリの様々なコンテナタイプを実装します。
各コンテナについて、適切な名前のクラスファイルを提出する。
名前空間は常に ft で、コンテナは ft::<container> を使ってテストされます。あなたは参照コンテナの構造を尊重する必要があります。もし、オーソドックスな正規形の一部が欠落しているならば、それを実装してはいけません。
注意点として、我々はC++98でコーディングしているので、コンテナの新機能はMUST NOT
は実装されますが、あらゆる古い機能（非推奨も含む）が期待されます。

第三章 必須項目

- 以下のコンテナを実装し、必要なファイル <container>.hpp を回してください。
- また、評価のためにすべてをテストするmain.cppを提供する必要があります。(例よりもっと先に進む必要があります!)
- 自分のコンテナだけを使ったバイナリと、STLコンテナを使って同じテストをしたバイナリを作成する必要があります。
- 出力とタイミングを比較する（最大で20倍遅くなることもある）。
- メンバー関数、非メンバー関数、オーバーロードを想定しています。
- ネーミングを尊重し、細部にまでこだわる。
- std::allocatorを使用する必要があります。
- 各コンテナの内部データ構造を正当化する必要があります（マップのために単純な配列を使用することはOKではありません）。
- コンテナがイテレータシステムを持つ場合、それを実装する必要があります。
- iterators_traits, reverse_iterator, enable_if, is_integral, equal/lexicographical compare, std::pair, std::make_pair は再実装する必要があります。
- https://www.cplusplus.com/ と https://cppreference.com/ を参考にすることができます。
- 標準コンテナで提供されるもの以上のパブリック関数を実装することはできません。それ以外はすべてプライベートかプロテクトでなければなりません。各公開関数・変数には、正当な理由が必要です。
- 非メンバーのオーバーロードでは、キーワードfriendを使用することができます。friendの各使用は
  は正当でなければならず、評価時に確認される。
- map::value_compare の実装では、friend というキーワードを使用することができる。

以下の容器とそれに付随する機能を提出する必要があります。

- ベクター
- 地図
- スタック

vectorの実装では、vector<bool>の特殊化をコード化することは必須ではありません。
スタックはベクタークラスをデフォルトの下敷きコンテナとして使用しますが、STLのような他のコンテナとの互換性が必要です。
STLコンテナは禁止されています。
STDライブラリの利用が許可されている。

第四章 ボーナス

パート

必須パートを終えた方は、ボーナスを回してみてください。

おまけで最後の容器を一つ。

- セット - ただし、今回はブラック - レッドのツリーが必須です。