vector 以外のコンテナについて

takashi ohya

2019/3/5

1. 内容について

この pdf では、vector 以外によく使われるコンテナの例として、std::list, std::map, std::set の 3 つについて簡易的に紹介する。

2. 各コンテナの紹介

2.1 std::list

std::vector は末尾の要素の追加・削除を行うことには向いているが、std::vector の要素はメモリ上に並んでいる必要があるため、末尾以外の要素の変更が行われる場合はその後ろにある要素を全てコピーする必要があり、非効率的である。それを避けたい場合、std::list を用いるのが推奨される。具体的には、std::vector の N 番目のデータ位置への挿入にかかる時間は $O(N)$ であるのに対して、std::list の任意位置への挿入は $O(1)$ である。一方、任意の位置へのアクセス（参照及び代入）に関しては std::vector が $O(1)$ であるのに対して std::list は $O(N)$ であるため、使い分けることが必要である。基本的な std::list の使い方は、src/day2/problem1/list/01_list.cpp に示した。

2.2 std::map

std::map はコンテナクラスの一つで、key と value のペアを要素として保持しており、key を指定することで高速に value を取り出すことが出来る機能を持っている。具体的には、配列を用いて key から value を取得するための処理時間は $O(N)$ であるが、std::map を用いると $O(\log(N))$ となる。std::unordered_map は更に高速で $O(1)$ であるが、key の順番を保持することが出来ないため、使い分ける必要がある。基本的な std::map の使い方は、src/day2/problem1/map/02_map.cpp に示した。

2.3 std::set

std::set は順序付けされた集合を意味するコンテナクラスである。データを std::set に追加すると、自動的に内部でソートが行われ格納されるため、データの追加・削除・検索時の処理速度が $O(\log(N))$ と比べて高速である。データの追加が動的に行われる場合に有用なクラスということが出来る。基本的な std::set の使い方は、src/day2/problem1/set/03_set.cpp に示した。

3. 参考文献

1) 矢吹太郎, 基礎からしっかり学ぶ C++ の教科書, 日経 BP 社, 119-142, 2017.
2) C++ コンテナクラス入門, 津田伸秀, http://vivi.dyndns.org/tech/cpp/container.html, 2019/3/5.