Popularne biblioteki C++

Biblioteka standardowa

1. Biblioteki „odziedziczone” z C

- <cassert> (assert.h)
- <cctype> (ctype.h)
- <cerrno> (errno.h)
- <cfenv> (fenv.h)
- <cfloat> (float.h)
- <cinttypes> (inttypes.h)
- <ciso646> (iso646.h)
- <climits> (limits.h)
- <clocale> (locale.h)
- <cmath> (math.h)
- <csetjmp> (setjmp.h)
- <csignal> (signal.h)
- <cstdarg> (stdarg.h)
- <cstdbool> (stdbool.h)
- <cstddef> (stddef.h)
- <cstdint> (stdint.h)
- <cstdio> (stdio.h)
- <cstdlib> (stdlib.h)
- <cstring> (string.h)
- <ctgmath> (tgmath.h)
- <ctime> (time.h)
- <cuchar> (uchar.h)
- <cwchar> (wchar.h)
- <cwctype> (wctype.h)

Biblioteka standardowa

2. Kontenery (STL)

- <array>
- <deque>
- <forward_list>
- <list>
- <map>
- <queue>
- <set>
- <stack>
- <unordered_map>
- <unordered_set>
- <vector>

Biblioteka standardowa

3. Operacje wejścia wyjścia (strumienie)

- <fstream>
- <iomanip>
- <ios>
- <iosfwd>
- <iostream>
- <istream>
- <ostream>
- <sstream>
- <streambuf>

Biblioteka standardowa

4. Wielowątkowość

- <atomic>
- <condition_variable>
- <future>
- <mutex>
- <thread>

Biblioteka standardowa

5. Inne

- <algorithm>
- <bitset>
- <chrono>
- <codecvt>
- <complex>
- <exception>
- <functional>
- <initializer_list>
- <iterator>
- <limits>
- <locale>
- <memory>
- <new>
- <numeric>
- <random>
- <ratio>
- <regex>
- <stdexcept>
- <string>
- <system_error>
- <tuple>
- <typeindex>
- <typeinfo>

www.boost.org:

- *Boost provides free peer-reviewed portable C++ source libraries.*

- *We emphasize libraries that work well with the C++ Standard Library. Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages both commercial and non-commercial use.*

- Boost to kolekcja plików nagłówkowych
- Funkcjonalność stopniowo przenoszona do samego języka (C++11, C++14, C++17)

Algorytmy

- foreach – BOOST_FOREACH makro dla prostej iteracji elementów sekwencyjnych
- graph – Ogólne komponenty i algorytmy dla grafów
- minmax – standardowe rozszerzenie biblioteki dla jednoczesnego obliczania min/max i min/max elementu
- range – Nowa infrastruktura generic algorithms będąca efektem nowej koncepcji iteratora
- string_algo – Biblioteka algorytmów pracujących na łańcuchach
- utility – Szablon funkcji klas next(), prior()

Programowanie współbieżne

- asio – Przenośne sieci i inne interfejsy I/O niskiego poziomu, w tym: sockets, timers, hostname resolution, socket iostreams, serial ports, file descriptors oraz Windows HANDLEs
- interprocess – pamięć współdzielona, pliki mapowania pamięci, współdzielenie między-procesowe mutexes, zmienne warunkowe, kontenery i wskaźniki
- MPI – biblioteka Message Passing Interface, do użytku przy programowaniu aplikacji obliczeń równoległych typu distributed-memory
- thread – Przenośny C++ multi-threading

Kontenery

- array – struktura kontenera (STL compliant container wrapper) dla tablic o stałym rozmiarze
- bimap – Mapowanie dwukierunkowe (maps)
- circular_buffer – kontener STL znany również jako bufor cykliczny
- dynamic_bitset – Wersja dynamicznie zmienna std::bitset
- graph – Ogólne komponenty i algorytmy dla grafów
- intrusive – Intrusywne contenery i algorytmy
- multi_array – Adaptery i Kontenery wielowymiarowe dla tablic przyległych danych
- multi_index – Kontenery z wielokrotnym interfejsem dostępu kompatybilnego z STL
- pointer container – Kontenery do zapisu pamięci dynamicznie alokowanej (heap – dla ułatwienia programowania zorientowanego obiektowo
- property map – Interfejsy definiowania koncepcyjnego mapujące obiekty kluczy z obiektami wartości
- unordered – nieuporządkowane kontenery asocjacyjne
- variant – Bezpieczny, ogólny, bazujący na stosie stack-based kontener z wariantami

Sprawdzanie poprawności i testowanie

- concept check – Ogólne narzędzie programowania
- static_assert – Statyczna asercja (asercja podczas kompilacji)
- test – Wsparcie dla prostych testów programu, testów full unit oraz monitorowania pracy programu

Struktury danych

- any – Bezpieczny, ogólny kontener dla pojedynczych wartości różnych typów
- bimap – Dwukierunkowe mapy
- compressed_pair – Optymalizacja pustego elementu
- fusion – Biblioteka przeznaczona do pracy z TUPLES, z różnymi kontenerami, algorytmami itp.
- multi_index – Kontenery z STL-kompatybilnym interfejsem wielodostępowym
- pointer container – Kontener do zapisu dynamicznie alokowanych obiektów polimorficznych, ułatwiający programowanie zorientowane obiektowo
- property tree – Drzewiasta struktura danych przeznaczona do przechowywania danych konfiguracyjnych
- tuple – Proste definicje funkcji zwracających dane złożone itp..
- uuid – Universally Unique Identifiers
- variant – Stabilny, ogólny, bazujący na stosie kontener wariantowy

- Programowanie uogólnione
- Grafy
- Wejście/wyjście
- Wspomaganie dla łączenia z programami napisanymi w języku Python
- Iteratory
- Matematyka i obliczenia numeryczne
- Zarządzanie pamięcią
- Parsery
- Metaprogramowanie za pomocą preprocesora
- Inteligentne wskaźniki z automatycznym zliczaniem referencji
- Łańcuchy znaków i przetwarzanie tekstów
- Metaprogramowanie z użyciem szablonów

boost::regex

- boost::regex_match

```
boost::regex e("(a?(ba)+\\d+)(tm)?");
std::string s("tmababa11tm");
if (boost::regex_match(s, e))
    …
```

boost::regex

- boost::regex_search

```
boost::regex e("(abc)?(def)(ghi)+");
std::string s("abcdefghighi defdefghi defghi");
std::string::const_iterator start, end;
start = s.begin();
end = s.end();
boost::match_results<std::string::const_iterator> matched;
while (boost::regex_search(start, end, matched, e))
{
    std::cout << " " << matched.str() << std::endl;
    start = matched[0].second;
}
```

boost::tuple

```
Tuple tup3(3, 4, 5, 'b');
boost::tie(a,b,c,d) = tup3;

int i = boost::get<0>(tup);
//tup.get<0>();
boost::get<1>(tup) = 5.4;

std::cout << tup << std::endl;
```

- **Qt Core**
- **Qt GUI**
- Qt Multimedia
- Qt Multimedia Widgets
- Qt Network
- Qt QML
- Qt Quick
- Qt Quick Controls
- Qt Quick Dialogs
- Qt Quick Layouts
- Qt SQL
- Qt Test
- **Qt Widget**

```
#include <QApplication>
#include <QPushButton>

int main(int argc, char **argv)
{
 QApplication app (argc, argv);

 QWidget window;
 window.setFixedSize(100, 50);

 QPushButton *button = new QPushButton("Hello World", &window);
 button->setGeometry(10, 10, 80, 30);

 window.show();
 return app.exec();
}
```