**Qset**

Cоздание и заполнение

#include <QCoreApplication>

#include <QTextStream>

#include <QSet>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QSet<int> set;

set.insert(10);

set.insert(20);

set.insert(30);

for(const auto &num : set)

{

out << num << " ";

}

out.flush();

return a.exec();

}

Проверка наличия элемента в QSet:

#include <QCoreApplication>

#include <QTextStream>

#include <QSet>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QSet<int> set;

set.insert(10);

set.insert(20);

set.insert(30);

out << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

out << Qt::endl;

if(set.contains(10))

{

out << "Element found in QSet" << Qt::endl;

} else

{

out << "Element not found in QSet" << Qt::endl;

}

out.flush();

return a.exec();

}

Удаление элемента из Qset и получение размера:

#include <QCoreApplication>

#include <QTextStream>

#include <QSet>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QSet<int> set;

set.insert(10);

set.insert(20);

set.insert(30);

out << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

int size = set.size();

out << Qt::endl << size << Qt::endl;

set.remove(20);

int newsize = set.size();

out << newsize << Qt::endl;

out.flush();

return a.exec();

}

Выполнение операции объединения с другим QSet:

#include <QCoreApplication>

#include <QTextStream>

#include <QSet>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QSet<int> set;

set.insert(10);

set.insert(20);

set.insert(30);

out << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

QSet<int> otherSet;

otherSet.insert(40);

otherSet.insert(50);

set.unite(otherSet);

out << Qt::endl << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

out.flush();

return a.exec();

}

Выполнение операции пересечения с другим QSet:

#include <QCoreApplication>

#include <QTextStream>

#include <QSet>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QSet<int> set;

set.insert(10);

set.insert(20);

set.insert(30);

out << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

QSet<int> otherSet;

otherSet.insert(10);

otherSet.insert(30);

set.intersect(otherSet);

out << Qt::endl << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

out.flush();

return a.exec();

}

Выполнение операции разности с другим QSet:

#include <QCoreApplication>

#include <QTextStream>

#include <QSet>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QSet<int> set;

set.insert(10);

set.insert(20);

set.insert(30);

out << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

QSet<int> otherSet;

otherSet.insert(20);

otherSet.insert(30);

set.subtract(otherSet);

out << Qt::endl << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

out.flush();

return a.exec();

}

Выполнение операции симметрической разности с другим QSet:

#include <QCoreApplication>

#include <QTextStream>

#include <QSet>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QSet<int> set;

set.insert(10);

set.insert(20);

set.insert(30);

out << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

QSet<int> otherSet;

otherSet.insert(30);

otherSet.insert(40);

set.unite(otherSet);

set.subtract(otherSet);

out << Qt::endl << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

out.flush();

return a.exec();

}

Проверка наличия пересечения с другим QSet:

#include <QCoreApplication>

#include <QTextStream>

#include <QSet>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QSet<int> set;

set.insert(10);

set.insert(20);

set.insert(30);

out << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

QSet<int> otherSet;

otherSet.insert(40);

otherSet.insert(50);

if (set.intersects(otherSet))

{

out << "Intersection found in QSet";

} else

{

out << "Intersection not found in QSet";

}

out.flush();

return a.exec();

}

Проверка на равенство двух QSet:

#include <QCoreApplication>

#include <QTextStream>

#include <QSet>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QSet<int> set;

set.insert(10);

set.insert(20);

set.insert(30);

out << "Elements: " << Qt::endl;

for(const auto &num : set)

{

out << num << " ";

}

QSet<int> otherSet;

otherSet.insert(10);

otherSet.insert(20);

if (set == otherSet)

{

qDebug() << "QSet is equal to other set";

} else

{

qDebug() << "QSet is not equal to other set";

}

out.flush();

return a.exec();

}

**Qmap**

Создание пустого Qmap и заполнение элементами:

#include <QCoreApplication>

#include <QTextStream>

#include <QMap>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QMap<QString, int> map;

map.insert("apple", 3);

map.insert("banana", 2);

map.insert("orange", 5);

for (const auto& key : map.keys())

{

const int value = map.value(key);

qDebug() << key << ":" << value;

}

out.flush();

return a.exec();

}

Действия над контейнерами

#include <QCoreApplication>

#include <QTextStream>

#include <QMap>

#include <QDebug>

#include <algorithm>

int main(int argc, char \*argv[])

{

QTextStream in(stdin);

QTextStream out(stdout);

QCoreApplication a(argc, argv);

QMap<QString, int> map;

map.insert("apple", 3);

map.insert("banana", 2);

map.insert("orange", 5);

for (const auto& key : map.keys())

{

const int value = map.value(key);

qDebug() << key << ":" << value;

}

// Проверка, пустой ли QMap:

bool isEmpty = map.isEmpty();

out << isEmpty << Qt::endl;

// Получение наибольшего ключа в QMap:

QString largestKey = map.lastKey();

out << largestKey << Qt::endl;

// Получение наименьшего ключа в QMap:

QString smallestKey = map.firstKey();

out << smallestKey << Qt::endl;

// Получение наибольшего значение в QMap:

// int largestValue = \*std::max\_element(map.values().begin(), map.values().end());

out << \*std::max\_element(map.begin(), map.end()) << Qt::endl;

// Получение наименьшего значение в QMap:

int smallestValue = \*std::min\_element(map.begin(), map.end());

out << smallestValue << Qt::endl;

out.flush();

return a.exec();

}