Вебинар 3. Коды. Контейнеры.

**Qvector**

Пример 1 – создание и вывод элементов вектора

#include <QCoreApplication>

#include <QVector>

#include <QTextStream>

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

{

QTextStream in(stdin);

QTextStream out(stdout);

QVector<int> numbers {1, 2, 3, 4, 5};

// Выввод элементов:

for (int i=0; i < numbers.size(); ++i)

{

out << numbers[i] << " ";

}

out.flush();

}

Пример 2 - Создание экземпляра QVector типа QString и добавление значений "John", "Jane", "Mike" с помощью оператора <<.

#include <QCoreApplication>

#include <QVector>

#include <QTextStream>

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

{

QCoreApplication a(argc, argv);

QTextStream in(stdin);

QTextStream out(stdout);

QVector<QString> names;

names << "John" << "Jane" << "Mike";

// Выввод элементов:

for (int i=0; i < names.size(); ++i)

{

out << names[i] << " ";

}

out.flush();

return a.exec();

}

Пример 3 - Создание экземпляра QVector типа double и инициализация его значениями 0.0 повторяющимися 24 раза.

#include <QCoreApplication>

#include <QVector>

#include <QTextStream>

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

{

QCoreApplication a(argc, argv);

QTextStream in(stdin);

QTextStream out(stdout);

QVector<double> temperatures(24, 0.0);

// Выввод элементов:

for (int i=0; i < temperatures.size(); ++i)

{

out << temperatures[i] << " ";

}

out.flush();

return a.exec();

}

Пример 4 – работа с вектором, элементами которого являются целые числа.

#include <QVector>

#include <QTextStream>

int main(void) {

QTextStream out(stdout);

// Создаем вектор, содержащий целочисленные значения

QVector<int> vals = {1, 2, 3, 4, 5};

// С помощью метода size() возвращаем размер вектора (количество элементов, содержащихся в нем)

out << "The size of the vector is: " << vals.size() << Qt::endl;

out << "The first item is: " << vals.first() << Qt::endl; // получаем первый элемент вектора

out << "The last item is: " << vals.last() << Qt::endl; // получаем последний элемент вектора

vals.append(6); // вставляем новый элемент в конец вектора

vals.prepend(0); // вставляем новый элемент в начало вектора

out << "Elements: ";

// Перебираем элементы вектора и выводим их на экран

for (int val : vals) {

out << val << " ";

}

out << Qt::endl;

return 0;

}

**Qlist**

Пример 1 – Создание списка целых чисел:

#include <QCoreApplication>

#include <QList>

#include <QTextStream>

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

{

int i;

QCoreApplication a(argc, argv);

QTextStream in(stdin);

QTextStream out(stdout);

QList<int> numbers;

numbers.append(10);

numbers.append(20);

numbers.append(30);

for (i=0; i < numbers.size(); ++i)

{

out << numbers.at(i) << " ";

}

out.flush();

return a.exec();

}

Пример 2 - Добавление элементов в список и извлечение их:

#include <QCoreApplication>

#include <QList>

#include <QTextStream>

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

{

int i;

QCoreApplication a(argc, argv);

QTextStream in(stdin);

QTextStream out(stdout);

QList<QString> names;

names << "Alice" << "Bob" << "Sam";

QString first = names.first(); // получение первого элемента списка

QString last = names.last(); // получение последнего элемента списка

for (i=0; i < names.size(); ++i)

{

out << names.at(i) << " ";

}

out.flush();

return a.exec();

}

Пример 3 - Перебор элементов списка с использованием итератора:

#include <QCoreApplication>

#include <QList>

#include <QTextStream>

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

{

int i;

QCoreApplication a(argc, argv);

QTextStream in(stdin);

QTextStream out(stdout);

QList<QString> fruits;

fruits << "apple" << "banana" << "cherry";

QListIterator<QString> iter(fruits);

while (iter.hasNext())

{

QString fruit = iter.next();

qDebug() << fruit;

}

out.flush();

return a.exec();

}

Пример 4 - Удаление элементов из списка:

#include <QCoreApplication>

#include <QList>

#include <QTextStream>

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

{

int i;

QCoreApplication a(argc, argv);

QTextStream in(stdin);

QTextStream out(stdout);

QList<int> numbers;

numbers << 1 << 2 << 3 << 4 << 5;

// вывод изначального контейнера

for (i=0; i< numbers.size(); ++i)

{

out << numbers.at(i) << " ";

}

out << Qt::endl;

out << "Delete..." << Qt::endl;

numbers.removeOne(3); // удаление элемента 3

numbers.removeAll(4); // удаление всех элементов со значением 4

// вывод отредактированного контейнера

for (i=0; i< numbers.size(); ++i)

{

out << numbers.at(i) << " ";

}

out.flush();

return a.exec();

}

**QStringList**

Пример 1 – создание, заполнение, размер контейнера, проверка на наличие элементов

#include <QCoreApplication>

#include <QList>

#include <QTextStream>

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

{

int i;

QCoreApplication a(argc, argv);

QTextStream in(stdin);

QTextStream out(stdout);

// Создание пустого списка:

QStringList list;

// Добавление элементов в список:

list << "apple" << "pear" << "orange";

// Получение количества элементов в списке:

int count = list.size();

out << "Size: " << count << Qt::endl;

// Получение элемента списка по индексу:

QString fruit = list.at(0); // fruit = "Яблоко"

out << fruit << Qt::endl;

// Проверка наличия элемента в списке:

bool containsFruit = list.contains("pear");

out << containsFruit;

out.flush();

return a.exec();

}

**Qqueue**

Пример 1 - Использование QQueue для FIFO (First In First Out):

#include <QCoreApplication>

#include <QQueue>

#include <QDebug>

#include <QTextStream>

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

{

int i;

QCoreApplication a(argc, argv);

QTextStream in(stdin);

QTextStream out(stdout);

QQueue<QString> names;

names.enqueue("Alice"); // добавление элементов в очередь

names.enqueue("Bob");

names.enqueue("Sofi");

while (!names.isEmpty())

{

qDebug() << names.dequeue(); // удаление элементов из очереди и их вывод

}

out.flush();

return a.exec();

}