1. Какие существуют способы обработки различных ошибок?

а) Ошибки программиста (алгоритмические) можно отлавливать с помощью assert (в режиме debug), отладчика или специально написанных тестов.

б) Runtime-ошибки можно обрабатывать с использованием

1) аварийного завершения программы (abort/terminate/exit).

2) условных операторов и циклов (if-else, while) (если этот вариант возможен, то он является наиболее предпочтительным).

3) прописывания и анализа кодов возврата функций.

4) С использованием механизма исключений из ядра языка c++.

2. В чем заключаются недостатки механизма кодов возврата?

Низкая читабельность кода (которую, правда, можно решить с помощью перечислений), неудобство поддержки и расширения программ, конвертации кодов возврата, пересечение с возвращаемыми значениями функций (решаемое с помощью глобальных переменных, дополнительных аргументов или возврата составных типов данных), наличие сущностей (а именно конструкторов, деструкторов и операторов), которые не возвращают значений или их набор фиксирован и ограничен.

3. Какими особенностями обладает механизм исключений?

1. Удобен и эффективен для обработки любых исключительных ситуаций, позволяет пропускать часть кода, если такие ситуации возникают.

2. Многоуровневость: throw может подняться в поисках catch на любое количество уровней вверх, пока не отыщет обработку catch-ем для себя (вплоть до завершения выполнения программы, если ничего не найдет).

3. Преобразования типов не выполняются, при этом в catch можно запускать новый throw.

4. Для чего используется спецификатор и оператор noexcept?

Спецификатор noexcept говорит, что функция не генерирует исключений, а если это все же произойдет, то немедленно запустится std::terminate(). Это позволяет компилятору генерировать более оптимальный код.

Оператор noexcept возвращает true, если функция-аргумент объявлена noexcept, и false в противном случае.

5. Как формулируются гарантии безопасности исключений?

1. Базовая гарантия (обязательно нужно соблюдать при написании) — при использовании механизма исключений обеспечиваем отсутствие утечек памяти и нарушений инвариантов.

2. Строгая гарантия — при возникновении ошибок обеспечиваем откат всех действий, которые привели к возникновению этих ошибок.

3. Гарантия отсутствия исключений - совсем не генерируем исключений.