

# Run-time Type Information Solutions

# typeid

- What does typeid() do? When could it be useful?
  - typeid() returns a `type_info` object containing information about the dynamic type of its argument
  - It is mainly used in comparisons, to determine whether two objects have the same dynamic type
- Write a simple program which uses typeid()

# type\_info

- Briefly describe type\_info
  - type\_info is a class which contains information about the dynamic type of an object
  - It has a name() member function which returns a C-style string
- Write a simple program which uses type\_info

# hash\_code

- Briefly describe hash\_code
  - hash\_code is a member function of type\_info
  - It returns a number which depends upon the dynamic type of the object
  - This number has the same value for all objects of the same dynamic type
- Write a simple program which uses hash\_code

# dynamic\_cast

- What safety features does `dynamic_cast` include?
  - The cast is only done if the dynamic type of the argument is the same as the type being cast to
  - For a pointer to base, a null pointer is returned on failure
  - For a reference to base, `std::bad_cast` exception is thrown
- Why are these safety features needed?
  - A base-to-derived conversion is potentially dangerous
  - If the resulting object is not the expected type, we may access data members or call member functions which do not exist, or which do not produce the expected result

# dynamic\_cast

- Write a simple program which uses dynamic\_cast with
  - A pointer to the base class
  - A reference to the base class
- What happens when the cast fails?
  - For the case of pointer to base, dynamic\_cast returns a null pointer
  - The program's behaviour is undefined
  - For a reference to base, dynamic\_cast throws std::bad\_cast on error