ASL Median and Selection

- The **median** algorithm returns the second of three arguments. The algorithm is stable, which is to say that if the arguments are in non-decreasing order then the identity of the returned element will be the second identity of the second argument.
- The **select()** algorithms are building blocks for other algorithms such as median() and clamp(). The general form of a select algorithm is:
  select_<argument_index>_<argument_ordering>()
  For example: select_1_ac(a, b, c) means "select the second element (index starts at zero) assuming that arguments ***a*** and ***b*** are supplied in non-decreasing order.“
  All of the select functions are stable.

ASL Median

- template<typename T , typename R >
  const T & median (const T &a, const T &b, const T &c, R r)
- template<typename T >
  T & median (T &a, T &b, T &c)
- template<typename T >
  const T & median (const T &a, const T &b, const T &c)
- template<typename T , typename R >
  T & median (T &a, T &b, T &c, R r)

ASL Selection

- template<typename T , typename R >
  const T & select_1 (const T &a, const T &b, const T &c, R r)
- template<typename T , typename R >
  T & select_1 (T &a, T &b, T &c, R r)
- template<typename T , typename R >
  T & select_1_ab (T &a, T &b, T &c, R r)
- template<typename T , typename R >
  const T & select_1_ab (const T &a, const T &b, const T &c, R r)
- template<typename T , typename R >
  const T & select_1_ac (const T &a, const T &b, const T &c, R r)
- template<typename T , typename R >
  T & select_1_ac (T &a, T &b, T &c, R r)

Implementation

- template <typename T, typename R>
  inline const T& median(const T& a, const T& b, const T& c,
  R r)
  { return select_1_3(a, b, c, boost::bind(r, _1, _2)); }
- template <typename T, typename R>
  inline T& select_1_3(T& a, T& b, T& c, R r)
  { return r(b, a) ? select_1_3_ab(b, a, c, r)
  : select_1_3_ab(a, b, c, r); }
- template <typename T, typename R>
  inline T& select_1_3_ab(T& a, T& b, T& c, R r)
  { assert(!r(b, a) && "WARNING (sparent) : a and b must be
  non-decreasing");
  return r(c, b) ? select_1_2(a, c, r) : b;
  }