

## Section 2.1 C++11

## auto Variables

```

thread_local auto localCounter = 0L; // long
static constexpr auto pi = 3.1415926535f; // float

```

Finally, **auto** variables may be declared in any location that allows declaring a variable supplied with an initializer with a single exception of **nonstatic data members**; see *Annoyances* — **auto** is disallowed for nonstatic data members on page 212:

```

// namespace scope
auto globalNamespaceVar = 3.; // double

namespace ns
{
    static auto nsNamespaceVar = "..."; // const char*
}

enum Status { /*...*/ };

int sendRequest();
Status responseStatus();
bool haveMoreWork();
void f()
{
    // block scope
    constexpr auto blockVar = 'a'; // char

    // condition of if, switch, and while statements
    if (auto rc = sendRequest()) { /*...*/ } // int
    switch (auto status = responseStatus()) { /*...*/ } // Status
    while (auto keepGoing = haveMoreWork()) { /*...*/ } // bool

    // init-statement of for loops
    std::vector<int> v;
    for (auto it = v.begin(); it != v.end(); ++it) // std::vector<int>::iterator
    { /*...*/ }

    // range declaration of range-based for loops
    for (const auto& constVal : v) { /*...*/ } // const int&
}

struct S
{
    // static data members
    static const auto k_CONSTANT = 11u; // const unsigned int
};

```