How we reason about procedural programs

Lisa Lippincott

```
void f()
   {
   }
   void f()
     {
☞    }
```

```
void f()
   {
    assert( true );
   }
   void f()
     {
☞     assert( true );
     }
   void f()
     {
      assert( true );
☞    }
```

```
void f()
   {
    claim true;
   }
```

```
void f()
  {
   bool b = true;
   claim b;
  }
  void f()
    {
☞    bool b = true;
     claim b;
    }
  void f()
    {
☞    bool b = true;
     claim b;
    }
```

```
   void f()
     {
☞     bool b = true;
      claim true /* b */;
     }
```

```
  void f()
    {
     bool b = true;
☞    claim true /* b */;
    }
  void f()
     {
      bool b = true;
      claim true /* b */;
☞    }
```

```
void f( bool b )
  {
   if ( b )
      claim b;
  }
```

```
void f( bool b )
   {
    if ( b )
        claim b;
   }
```

```
  void f( bool b )
    {
☞    if ( b )
        claim b;
    }
```

```
void f( bool b )
   {
☞   if ( false /* b */ )
       claim false /* b */;
   }
```

```
void f( bool b )
   {
☞   if ( true /* b */ )
       claim true /* b */;
   }
```

```
  void f( bool b )
    {
☞

    }
```

```
  void f( bool b )
    {
☞          claim true /* b */;
    }
  void f( bool b )
     {


☞    }
```

```
  void f( bool b )
     {

           claim true /* b */;
☞    }
```

```
void f( bool b )
   {
    if ( !b )
        b = true;
    claim b;
   }
```

```
void f( bool b )
  {
   if ( !b )
     {
      b = true;
      claim b;
     }
   else
      claim b;
  }
```

```
void f( bool b )
  {
☞  if ( !b )
     {
      b = true;
      claim b;
     }
   else
      claim b;
  }
  void f( bool b )
    {
☞    if ( b ? false : true )
        {
         b = true;
         claim b;
        }
     else
         claim b;
    }
```

```
   void f( bool b )
     {
☞     if ( false /* b */ ? false : true )
        {
         b = true;
         claim b;
        }
      else
         claim false /* b */;
     }
```

```
   void f( bool b )
     {
☞     if ( true /* b */ ? false : true )
        {
         b = true;
         claim b;
        }
      else
         claim true /* b */;
     }
```

```
  void f( bool b )
    {
☞    if ( true /* !b */ )
       {
        b = true;
        claim b;
       }
     else
        claim false /* b */;
    }
```

```
  void f( bool b )
    {
☞    if ( false /* !b */ )
       {
        b = true;
        claim b;
       }
     else
        claim true /* b */;
    }
  void f( bool b )
    {
☞

        b = true;
        claim b;



    }
```

```
  void f( bool b )
    {
☞





        claim true /* b */;
    }
```

```
void f( bool b )
  {

☞     b = true;
      claim true /* b */;



  }
```

```
void f( bool b )
  {
☞




      claim true /* b */;
  }
```

```
void f( bool b )
  {


      b = true;
☞     claim true /* b */;



  }
```

```
void f( bool b )
  {




☞     claim true /* b */;
  }
```

```
void f( bool b )
  {



     b = true;
     claim true /* b */;



☞  }
```

```
void f( bool b )
  {






       claim true /* b */;
☞  }
```

```
void f( int a, int b )
  {
   if ( a + 1 < b )
      claim a + 1 < b;
  }
```

```
void f( int a, int b )
  {
   auto t0 = a + 1;
   auto t1 = ( t0 < b );
   if ( t1 )
        {
         auto t2 = a + 1;
         auto t3 = ( t2 < b );
         claim t3;
        }
  }
```

```
void f( int a, int b )
  {
   auto t0 = a + 1;
   auto t1 = ( t0 < b );
   if ( t1 )
        {
         auto t2 = a + 1;
         auto t3 = ( t2 < b );
         claim t3;
        }
  }
```

```
void f( int a, int b )
  {
☞  auto t0 = a + 1;
   auto t1 = ( t0 < b );
   if ( t1 )
         {
          auto t2 = a + 1;
          auto t3 = ( t2 < b );
          claim t3;
         }
  }
```

```
   void f( int a, int b )
     {
☞     auto t0 = a + 1;
      auto t1 = ( t0 < b );
      if ( t1 )
            {
             auto t2 = a + 1;
             auto t3 = ( t2 < b );
             claim t3;
            }
     }
```

```
void f( int a, int b )
  {
☞  auto t0 = a + 1;
   auto t1 = ( t0 < b );
   if ( t1 )
        {
         auto t2 = a + 1;
         auto t3 = ( t2 < b );
         claim t3;
        }
  }
```

```
void f( int a, int b )
  {
   auto t0 = a + 1;
☞  auto t1 = ( t0 < b );
   if ( t1 )
        {
         auto t2 = a + 1;
         auto t3 = ( t2 < b );
         claim t3;
        }
  }
```

```
   void f( int a, int b )
     {
      auto t0 = a + 1;
☞     auto t1 = ( t0 < b );
      if ( t1 )
           {
            auto t2 = a + 1;
            auto t3 = ( t2 < b );
            claim t3;
           }
     }
```

```
  void f( int a, int b )
    {
     auto t0 = a + 1;
     auto t1 = ( t0 < b );
☞    if ( t1 )
          {
           auto t2 = a + 1;
           auto t3 = ( t2 < b );
           claim t3;
          }
    }
```

```
   void f( int a, int b )
     {
      auto t0 = a + 1;
      auto t1 = ( t0 < b );
☞     if ( false /* t1 */ )
           {
            auto t2 = a + 1;
            auto t3 = ( t2 < b );
            claim false /* t3 */;
           }
     }
```

```
   void f( int a, int b )
     {
      auto t0 = a + 1;
      auto t1 = ( t0 < b );
☞     if ( true /* t1 */ )
           {
            auto t2 = a + 1;
            auto t3 = ( t2 < b );
            claim true /* t3 */;
           }
     }
```

```
void f( int a, int b )
  {
   auto t0 = a + 1;
   auto t1 = ( t0 < b );

☞


  }
```

```
void f( int a, int b )
  {
   auto t0 = a + 1;
   auto t1 = ( t0 < b );

☞               {
                 auto t2 = a + 1;
                 auto t3 = ( t2 < b );
                 claim true /* t3 */;
                }
  }
```

```
void f( int a, int b )
  {
   auto t0 = a + 1;
   auto t1 = ( t0 < b );



☞



  }
```

```
void f( int a, int b )
  {
   auto t0 = a + 1;
   auto t1 = ( t0 < b );

         {
☞         auto t2 = a + 1;
          auto t3 = ( t2 < b );
          claim true /* t3 */;
         }
  }
```

```
void f( int a, int b )
  {
   auto t0 = a + 1;
   auto t1 = ( t0 < b );



☞



  }
```

```
void f( int a, int b )
  {
   auto t0 = a + 1;
   auto t1 = ( t0 < b );

        {
         auto t2 = a + 1;
☞        auto t3 = ( t2 < b );
         claim true /* t3 */;
        }
  }
void f( int a, int b )
  {
   auto t0 = a + 1;
   auto t1 = ( t0 < b );





☞



        }
```

```
  void f( int a, int b )
     {
      auto t0 = a + 1;
      auto t1 = ( t0 < b );

           {
            auto t2 = a + 1;
            auto t3 = ( t2 < b );
☞           claim true /* t3 */;
           }
     }
```

```
void f( int a, int b )
   {
    auto t0 = a + 1;
    auto t1 = ( t0 < b );







☞  }
```

```
void f( int a, int b )
   {
    auto t0 = a + 1;
    auto t1 = ( t0 < b );

         {
          auto t2 = a + 1;
          auto t3 = ( t2 < b );
          claim true /* t3 */;
         }
☞  }
```

```
int f( int a, int b )
  {
   if ( a < b )
      return -1;

   if ( a > b )
      return 1;




   claim a == b;
   return 0;
  }
```

```
int f( int a, int b )
  {
   if ( a < b )
      return -1;

   if ( a > b )
      return 1;

   posit a < b  ll  a == b  ll  a > b;

   claim a == b;
   return 0;
  }
```

```
int f( int a, int b )
  {
   if ( a < b )
       return -1;

   if ( a > b )
       return 1;

   posit a < b  ||  a == b  ||  a > b;

   claim a == b;
   return 0;
  }
```

```
int f( int a, int b )
  {
☞  if ( a < b )
       return -1;

   if ( a > b )
       return 1;

   posit a < b || a == b || a > b;

   claim a == b;
   return 0;
  }
```

```
int f( int a, int b )
  {
☞  if ( a < b )
       return -1;

    if ( a > b )
       return 1;

    posit a < b || a == b || a > b;

    claim a == b;
    return 0;
  }
```

```
int f( int a, int b )
 {
☞ if ( true /* a < b */ )
      return -1;

  if ( a > b )
      return 1;

  posit true /*a<b*/ || a == b || a > b;

  claim a == b;
  return 0;
 }
```

```
int f( int a, int b )
 {
☞ if ( false /* a < b */ )
      return -1;

  if ( a > b )
      return 1;

  posit false /*a<b*/ || a == b || a > b;

  claim a == b;
  return 0;
 }
```

```
  int f( int a, int b )
    {
☞
        return -1;






    }
```

```
  int f( int a, int b )
    {
☞


      if ( a > b )
          return 1;

      posit a == b || a > b;

      claim a == b;
      return 0;
    }
```

```
int f( int a, int b )
  {



☞  if ( a > b )
        return 1;

     posit a == b || a > b;

     claim a == b;
     return 0;
  }
```

```
int f( int a, int b )
  {

☞   if ( a > b )
        return 1;

    posit a == b  || a > b;

    claim a == b;
    return 0;
  }
```

```
int f( int a, int b )
  {



☞  if ( true /* a > b */ )
        return 1;

    posit a == b || true /*a > b */;

    claim a == b;
    return 0;
  }
```

```
int f( int a, int b )
  {



☞  if ( false /* a > b */ )
        return 1;

    posit a == b || false /* a > b */;

    claim a == b;
    return 0;
  }
```

```
int f( int a, int b )
  {


☞
       return 1;



  }
```

```
int f( int a, int b )
  {


☞

     posit a == b || false;

     claim a == b;
     return 0;
  }
```

```
int f( int a, int b )
  {



☞   posit a == b  ||  false;

    claim a == b;
    return 0;
  }
```

```
int f( int a, int b )
  {

☞   posit a == b || false;

    claim a == b;
    return 0;
  }
```

```
int f( int a, int b )
  {




☞    posit true /* a == b */  ||  false;

     claim true /* a == b */;
     return 0;
  }
```

```
int f( int a, int b )
  {




☞    posit false /* a == b */  ||  false;

     claim false /* a == b */;
     return 0;
  }
```

```
 int f( int a, int b )
   {



☞   posit true /* a == b */;

    claim true /* a == b */;
    return 0;
   }
```

```
 int f( int a, int b )
   {



☞   posit false;

    claim false;
    return 0;
   }
```

```
  int f( int a, int b )
    {





☞



     return 0;
    }
```

```
  int f( int a, int b )
    {





☞    posit false;

     claim false;
     return 0;
    }
```

```
int f( int a, int b )
  {

☞

   return 0;
  }
```

```
void f( int a, int b )
  {
   if ( a == b )
      assert( a + 1 == b + 1 );
  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {
      if ( a == b )
         substitutable( a, b );
     }

   if ( e )
     {
      posit a + 1 == a + 1;
      claim a + 1 == b + 1;
     }
  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {
      if ( a == b )
         substitutable( a, b );
     }

   if ( e )
     {
      posit a + 1 == a + 1;
      claim a + 1 == b + 1;
     }
  }
```

```
void f( int a, int b )
  {
☞  bool e = ( a == b );
   posit
     {
      if ( a == b )
         substitutable( a, b );
     }

   if ( e )
     {
      posit a + 1 == a + 1;
      claim a + 1 == b + 1;
     }
  }
```

```
   void f( int a, int b )
     {
☞     bool e = ( a == b );
      posit
        {
         if ( a == b )
            substitutable( a, b );
        }

      if ( e )
        {
         posit a + 1 == a + 1;
         claim a + 1 == b + 1;
        }
     }
```

```
  void f( int a, int b )
    {
     bool e = ( a == b );
☞    posit
       {
        if ( a == b )
           substitutable( a, b );
       }

     if ( e )
       {
        posit a + 1 == a + 1;
        claim a + 1 == b + 1;
       }
    }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {
☞     if ( a == b )
         substitutable( a, b );
     }

   if ( e )
     {
      posit a + 1 == a + 1;
      claim a + 1 == b + 1;
     }
  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {
☞      if ( false /* a == b */ )
          substitutable( a, b );
     }

   if ( false /* e */ )
     {
      posit a + 1 == a + 1;
      claim a + 1 == b + 1;
     }
  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {
☞      if ( true /* a == b */ )
          substitutable( a, b );
     }

   if ( true /* e */ )
     {
      posit a + 1 == a + 1;
      claim a + 1 == b + 1;
     }
  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {
☞

     }





  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {
☞        substitutable( a, b );
     }




      posit a + 1 == a + 1;
      claim a + 1 == b + 1;

  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {

☞
     }









  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {

☞          substitutable( a, b );
     }




     posit a + 1 == a + 1;
     claim a + 1 == b + 1;


  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {


     }



☞



  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {

         substitutable( a, b );
     }



☞       posit a + 1 == a + 1;
        claim a + 1 == b + 1;


  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {



     }



☞



  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {

         substitutable( a, b );
     }



☞      posit a + 1 == a + 1;
       claim a + 1 == b + 1;


  }
```

```
void f( int a, int b )
   {
    bool e = ( a == b );
    posit
      {


      }


☞


   }
```

```
void f( int a, int b )
   {
    bool e = ( a == b );
    posit
      {

           substitutable( a, b );
      }


☞       posit a + 1 == a + 1;
        claim a + 1 == b + 1;

   }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {


     }



☞



  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {

          substitutable( a, b );
     }



☞       posit a + 1 == a + 1;
        claim a + 1 == b + 1;


  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {


     }




☞




  }
```

```
void f( int a, int b )
  {
   bool e = ( a == b );
   posit
     {

         substitutable( a, b );
     }




☞        posit true /* a + 1 == a + 1 */;
         claim true /* a + 1 == b + 1 */;


  }
```

```
void f( int a, int b )
   {
    bool e = ( a == b );
    posit
       {


       }






☞


   }
```

```
void f( int a, int b )
   {
    bool e = ( a == b );
    posit
       {

          substitutable( a, b );
       }





        posit true /* a + 1 == a + 1 */;
☞       claim true /* a + 1 == b + 1 */;

   }
```

```
  void f( int a, int b )
    {
     bool e = ( a == b );
     posit
       {


       }





☞   }
```

```
  void f( int a, int b )
    {
     bool e = ( a == b );
     posit
       {

           substitutable( a, b );
       }




         posit true /* a + 1 == a + 1 */;
         claim true /* a + 1 == b + 1 */;

☞   }
```

```
void f( int a, int b )
  {
   int a1 = a + 1;
   int b1 = b + 1;

   if ( a == b )
      assert( a1 == b1 );
  }
```

```
void f( int a, int b )
  {
   int a1 = a + 1;
   int b1 = b + 1;
   bool e = ( a == b );
   posit
      {
       if ( a == b )
         substitutable( a, b );
      }

   if ( e )
     {
      posit a1 == a1;
      assert( a1 == b1 );
     }
  }
```

```
void f( int a, int b )
  {
   int a1 = a + 1;
   int b1 = b + 1;
   bool e = ( a == b );
   posit
      {
       if ( a == b )
         substitutable( a, b );
      }

   if ( e )
     {
      posit a1 == a1;
      assert( a1 == b1 );
     }
  }
```

```
void f( int a, int b )
  {
☞  int a1 = a + 1;
   int b1 = b + 1;
   bool e = ( a == b );
   posit
      {
       if ( a == b )
          substitutable( a, b );
      }

   if ( e )
     {
      posit a1 == a1;
      assert( a1 == b1 );
     }
  }
void f( int a, int b )
  {
   ☞    int a1 = a + 1;
   int b1 = b + 1;
   bool e = ( a == b );
   posit
      {
       if ( a == b )
          substitutable( a, b );
      }

   if ( e )
     {
      posit a1 == a1;
      assert( a1 == b1 );
     }
  }
```

```
 void f( int a, int b )
   {
☞   int a1 = a + 1;
    int b1 = b + 1;
    bool e = ( a == b );
    posit
       {
        if ( a == b )
          substitutable( a, b );
       }

    if ( e )
      {
       posit a1 == a1;
       assert( a1 == b1 );
      }
   }
```

```
  void f( int a, int b )
    {
     int a1 = a + 1;
☞    int b1 = b + 1;
     bool e = ( a == b );
     posit
        {
         if ( a == b )
           substitutable( a, b );
        }

     if ( e )
       {
        posit a1 == a1;
        assert( a1 == b1 );
       }
    }
```

```
void f( int a, int b )
  {
   int a1 = a + 1;
☞  int b1 = b + 1;
   bool e = ( a == b );
   posit
      {
       if ( a == b )
         substitutable( a, b );
      }

   if ( e )
     {
      posit a1 == a1;
      assert( a1 == b1 );
     }
  }
void f( int a, int b )
  {
   int a1 = a + 1;
   int b1 = b + 1;
   ☞    bool e = ( a == b );
   posit
      {
       if ( a == b )
         substitutable( a, b );
      }

   if ( e )
     {
      posit a1 == a1;
      assert( a1 == b1 );
     }
  }
  void f( int a, int b )
    {
     int a1 = a + 1;
     int b1 = b + 1;
   ☞      bool e = ( a == b );
     posit
         {
          if ( a == b )
            substitutable( a, b );
         }

     if ( e )
        {
         posit a1 == a1;
         assert( a1 == b1 );
        }
    }
```

```
   void f( int a, int b )
     {
      int a1 = a + 1;
      int b1 = b + 1;
      bool e = ( a == b );
☞     posit
         {
          if ( a == b )
            substitutable( a, b );
         }

      if ( e )
        {
         posit a1 == a1;
         assert( a1 == b1 );
        }
     }
```

```
  void f( int a, int b )
    {
     int a1 = a + 1;
     int b1 = b + 1;
     bool e = ( a == b );
     posit
        {
☞        if ( a == b )
           substitutable( a, b );
        }

     if ( e )
       {
        posit a1 == a1;
        assert( a1 == b1 );
       }
    }
```

```
void f( int a, int b )
  {
   int a1 = a + 1;
   int b1 = b + 1;
   bool e = ( a == b );
   posit
      {
☞      if ( false /* a == b */ )
         substitutable( a, b );
      }

   if ( false /* e */ )
     {
      posit a1 == a1;
      assert( a1 == b1 );
     }
  }
```

```
void f( int a, int b )
  {
   int a1 = a + 1;
   int b1 = b + 1;
   bool e = ( a == b );
   posit
      {
☞      if ( true /* a == b */ )
         substitutable( a, b );
      }

   if ( true /* e */ )
     {
      posit a1 == a1;
      assert( a1 == b1 );
     }
  }
  void f( int a, int b )
    {
     int a1 = a + 1;
     int b1 = b + 1;
     bool e = ( a == b );
     posit
       {

☞

       }





    }
```

```
  void f( int a, int b )
    {
     int a1 = a + 1;
     int b1 = b + 1;
     bool e = ( a == b );
     posit
       {

☞         substitutable( a, b );
       }



       posit a1 == a1;
       assert( a1 == b1 );

    }
```

```
void f( int a, int b )
  {
   int a1 = a + 1;
   int b1 = b + 1;
   bool e = ( a == b );
   posit
      {

☞
      }






  }
```

```
void f( int a, int b )
  {
   int a1 = a + 1;
   int b1 = b + 1;
   bool e = ( a == b );
   posit
      {

☞         substitutable( a, b );
      }




      posit a1 == a1;
      assert( a1 == b1 );

  }
```

```
void f( int a, int b )
  {
   if ( a == b )
     {}

   int a1 = a + 1;
   int b1 = b + 1;

   if ( a == b )
      claim a1 == b1;
  }
```

```
void f( int a, int b )
  {
   claim a == b ll a != b;


   int a1 = a + 1;
   int b1 = b + 1;

   if ( a == b )
      claim a1 == b1;
  }
```

```
void f( int a, int b )
  {
   int a1 = a + 1;

   claim a == b ll a != b;

   int b1 = b + 1;

   if ( a == b )
       claim a1 == b1;
  }
```

```
void f()
  {
   int i = 0;
   while ( i != 2 )
      i = i + 1;
  }
```

```
void f()
  {
   claim 0 + 1 + 1 == 2;

   int i = 0;
   while ( i != 2 )
      i = i + 1;
  }
```

```
void f()
  {
☞  claim 0 + 1 + 1 == 2;

   int i = 0;
   if ( i == 2 ) goto B;
   i = i + 1;
   while ( i == 2 )
      i = i + 1;
B: ;
  }
```

```
void f()
  {
☞   claim 0 + 1 + 1 == 2;

    int i = 0;
    if ( i == 2 ) goto B;
    i = i + 1;
    while ( i == 2 )
        i = i + 1;
B: ;
  }
```

```
void f()
  {
☞   claim 0 + 1 + 1 == 2;

    int i = 0;
    if ( i == 2 ) goto B;
    i = i + 1;
    if ( i == 2 ) goto B;
    i = i + 1;
    while ( i == 2 )
      i = i + 1;
B: ;
  }
```

```
void f()
  {
☞  claim 0 + 1 + 1 == 2;

   int i = 0;
   if ( i == 2 ) goto B;
  i = i + 1;
   if ( i == 2 ) goto B;
  i = i + 1;
   if ( i == 2 ) goto B;
   i = i + 1;
   while ( i != 2 )
      i = i + 1;
B: ;
  }
```

```
   void f()
     {
☞     claim 0 + 1 + 1 == 2;

      int i = 0;
      if ( i == 2 ) goto B;
      i = i + 1;
      if ( i == 2 ) goto B;
      i = i + 1;
      if ( i == 2 ) goto B;
      i = i + 1;
      while ( i != 2 )
         i = i + 1;
   B: ;
     }
```

```
void f()
  {
☞  claim 0 + 1 + 1 == 2;

   int i = 0;
   if ( i == 2 ) goto B;
   i = i + 1;
   if ( i == 2 ) goto B;
   i = i + 1;
   if ( i == 2 ) goto B;
   i = i + 1;
   while ( i != 2 )
     i = i + 1;
B: ;
  }
```

```
void f()
  {
☞   claim true /* 0 + 1 + 1 == 2 */;

    int i = 0;
    if ( i == 2 ) goto B;
    i = i + 1;
    if ( i == 2 ) goto B;
    i = i + 1;
    if ( true /* i == 2 */ ) goto B;
    i = i + 1;
    while ( i != 2 )
       i = i + 1;
B: ;
  }
```

```
void f()
  {
   claim true /* 0 + 1 + 1 == 2 */;
☞

   int i = 0;
   if ( i == 2 ) goto B;
   i = i + 1;
   if ( i == 2 ) goto B;
   i = i + 1;




B: ;
  }
```

```
void f( unsigned int n )
  {
   unsigned int i = 0;
   while ( i != n )
      i = i + 1;
  }
```

```
void f( unsigned int n )
  {
   posit
     {
      unsigned int i = 0;
      while ( i != n )
         i = i + 1;
     }
  }
```

```
void f( unsigned int n )
  {
   posit
     {
      unsigned int i = 0;
      while ( i != n )
         i = i + 1;
     }

   unsigned int j = 0;
   while ( j != n )
      j = j + 1;
  }
```

```
void f( unsigned int n )
  {
   posit
     {
      unsigned int i = 0;
      while ( i != n )   i = i + 1;
     }



   unsigned int j = 0;
   while ( j != n )   j = j + 1;
  }
```

```
void f( unsigned int n )
  {
   posit
     {
      unsigned int i = 0;
      while ( i != n )   i = i + 1;
     }



   unsigned int j = 0;
   while ( j != n )   j = j + 1;
  }
void f( unsigned int n )
  {
   ☞   posit
     {
      unsigned int i = 0;
      while ( i != n )   i = i + 1;
     }




   unsigned int j = 0;
   while ( j != n )   j = j + 1;
  }
void f( unsigned int n )
  {
   posit
     {
      ☞       unsigned int i = 0;
      while ( i != n )   i = i + 1;
     }



   unsigned int j = 0;
   while ( j != n )   j = j + 1;
  }
```

```
void f( unsigned int n )
  {
   posit
     {
☞     unsigned int i = 0;
      if ( i == n ) goto X;
      i = i + 1;
      while ( i != n )   i = i + 1;
X:;  }



   unsigned int j = 0;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
☞      if ( i == n ) goto X;
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;   }



   unsigned int j = 0;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
      unsigned int i = 0;
☞     if ( i == n ) goto X;
      i = i + 1;
      while ( i != n )   i = i + 1;
X:;   }
```

```
   unsigned int j = 0;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
  void f( unsigned int n )
    {
     posit
       {
        unsigned int i = 0;
☞       if ( false /* i == n */ ) goto X;
        i = i + 1;
        while ( i != n )   i = i + 1;
  X:;   }

     unsigned int j = 0;
     if ( false /* j == n */ ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
  Y:;}
```

```
  void f( unsigned int n )
    {
     posit
       {
        unsigned int i = 0;
☞       if ( true /* i == n */ ) goto X;
        i = i + 1;
        while ( i != n )   i = i + 1;
  X:;   }

     unsigned int j = 0;
     if ( true /* j == n */ ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
  Y:;}
```

```
   void f( unsigned int n )
     {
      posit
        {
          unsigned int i = 0;
☞         if ( false /* i == n */ ) goto X;
          i = i + 1;
          while ( i != n )   i = i + 1;
   X:;   }
```

```
      unsigned int j = 0;
      if ( false /* j == n */ ) goto Y;
      j = j + 1;
      while ( j != n )   j = j + 1;
   Y:;}
```

```
   void f( unsigned int n )
     {
      posit
        {
          unsigned int i = 0;
☞         if ( true /* i == n */ ) goto X;
          i = i + 1;
          while ( i != n )   i = i + 1;
   X:;   }
```

```
      unsigned int j = 0;
      if ( true /* j == n */ ) goto Y;
      j = j + 1;
      while ( j != n )   j = j + 1;
   Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;

☞
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;   }



    unsigned int j = 0;

    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;

☞



X:;   }



    unsigned int j = 0;




Y:;}
```

```
  void f( unsigned int n )
    {
     posit
       {
         unsigned int i = 0;

☞        i = i + 1;
         while ( i != n )   i = i + 1;
  X:;   }




     unsigned int j = 0;

     j = j + 1;
     while ( j != n )   j = j + 1;
  Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
      unsigned int i = 0;

☞     i = i + 1;
      while ( i != n )   i = i + 1;
X:;   }



   unsigned int j = 0;

   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
  void f( unsigned int n )
    {
     posit
       {
         unsigned int i = 0;
☞        i = i + 1;
         while ( i != n )   i = i + 1;
  X:;  }




     unsigned int j = 0;
     j = j + 1;
     while ( j != n )   j = j + 1;
  Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
      unsigned int i = 0;
☞     i = i + 1;
      while ( i != n )   i = i + 1;
X:;  }




     unsigned int j = 0;
     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
    posit
      {
        unsigned int i = 0;
☞      i = i + 1;
        if ( i == n ) goto X;
        i = i + 1;
        while ( i != n )   i = i + 1;
X:;   }


    unsigned int j = 0;
   j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;
☞      if ( i == n ) goto X;
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;  }
```

```
   unsigned int j = 0;
  j = j + 1;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
   void f( unsigned int n )
     {
      posit
        {
          unsigned int i = 0;
          i = i + 1;
☞         if ( i == n ) goto X;
          i = i + 1;
          while ( i != n )   i = i + 1;
   X:;  }


      unsigned int j = 0;
     j = j + 1;
      if ( j == n ) goto Y;
      j = j + 1;
      while ( j != n )   j = j + 1;
   Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;
☞      if ( false /* i == n */ ) goto X;
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;   }
```

```
    unsigned int j = 0;
   j = j + 1;
    if ( false /* j == n */ ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;
☞      if ( true /* i == n */ ) goto X;
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;   }
```

```
    unsigned int j = 0;
   j = j + 1;
    if ( true /* j == n */ ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;
☞      if ( false /* i == n */ ) goto X;
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;   }


     unsigned int j = 0;
    j = j + 1;
     if ( false /* j == n */ ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;
☞      if ( true /* i == n */ ) goto X;
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;   }


     unsigned int j = 0;
    j = j + 1;
     if ( true /* j == n */ ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;

☞
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;   }



     unsigned int j = 0;
    j = j + 1;

     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;

☞


X:;   }



     unsigned int j = 0;
    j = j + 1;



Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;

☞      i = i + 1;
       while ( i != n )   i = i + 1;
X:;   }



     unsigned int j = 0;
    j = j + 1;

     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
      unsigned int i = 0;
      i = i + 1;

☞     i = i + 1;
      while ( i != n )   i = i + 1;
X:;   }



   unsigned int j = 0;
  j = j + 1;

   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
   {
     posit
       {
         unsigned int i = 0;
         i = i + 1;
☞        i = i + 1;
         while ( i != n )   i = i + 1;
X:;    }



     unsigned int j = 0;
    j = j + 1;
     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

```
   void f( unsigned int n )
     {
      posit
        {
         unsigned int i = 0;
         i = i + 1;
☞        i = i + 1;
         if ( i == n ) goto X;
         i = i + 1;
         while ( i != n )   i = i + 1;
   X:;   }

      unsigned int j = 0;
      j = j + 1;
      j = j + 1;
      if ( j == n ) goto Y;
      j = j + 1;
      while ( j != n )   j = j + 1;
   Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
      unsigned int i = 0;
      i = i + 1;
☞     i = i + 1;
      if ( i == n ) goto X;
      i = i + 1;
      while ( i != n )   i = i + 1;
X:;   }

   unsigned int j = 0;
   j = j + 1;
   j = j + 1;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
 void f( unsigned int n )
   {
    posit
      {
        unsigned int i = 0;
        i = i + 1;
        i = i + 1;
☞       if ( i == n ) goto X;
        i = i + 1;
        while ( i != n )   i = i + 1;
 X:;  }

    unsigned int j = 0;
    j = j + 1;
    j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
 Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;

☞      if ( i == n ) goto X;
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;  }


   unsigned int j = 0;
  j = j + 1;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;
       i = i + 1;
☞      if ( i == n ) goto X;
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;  }


   unsigned int j = 0;
   j = j + 1;
  j = j + 1;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;

☞      if ( i == n ) goto X;
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;   }

     unsigned int j = 0;
    j = j + 1;
     if ( j == n ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

```
void f( unsigned int n )
  {
   posit
     {
       unsigned int i = 0;
       i = i + 1;
       i = i + 1;
☞      if ( i == n ) goto X;
       i = i + 1;
       while ( i != n )   i = i + 1;
X:;   }

     unsigned int j = 0;
     j = j + 1;
    j = j + 1;
     if ( j == n ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

✻
```
if ( i == n ) goto X;
i = i + 1;
```

One iteration

```
☞  unsigned int j = 0;
   j = j + 1;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

Two iterations

```
☞  unsigned int j = 0;
   j = j + 1;
   j = j + 1;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
* if ( i == n ) goto X;
  i = i + 1;
```

One iteration

```
    unsigned int j = 0;
☞   j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

Two iterations

```
    unsigned int j = 0;
☞   j = j + 1;
   j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

✻
```
if ( i == n ) goto X;
i = i + 1;
```

Two iterations

```
     unsigned int j = 0;
☞    j = j + 1;
     j = j + 1;
     if ( j == n ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

```
✻ if ( i == n ) goto X;
  i = i + 1;
```

Two iterations

```
☞   unsigned int j = 0;
    j = j + 1;
    j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
  Y:;}
```

Three iterations

```
☞   unsigned int j = 0;
    j = j + 1;
    j = j + 1
    j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
  Y:;}
```

Exiting after two iterations

```
☞   j = j + 1;
    j = j + 1;
  Y:;}
```

✻
```
if ( i == n ) goto X;
i = i + 1;
```

Two iterations

```
   unsigned int j = 0;
☞  j = j + 1;
  j = j + 1;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

Three iterations

```
   unsigned int j = 0;
☞  j = j + 1;
   j = j + 1
  j = j + 1;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
Y:;}
```

```
✻ if ( i == n ) goto X;
  i = i + 1;
```

Two iterations

```
   unsigned int j = 0;
   j = j + 1;
☞  j = j + 1;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
 Y:;}
```

Three iterations

```
   unsigned int j = 0;
   j = j + 1;
☞  j = j + 1
  j = j + 1;
   if ( j == n ) goto Y;
   j = j + 1;
   while ( j != n )   j = j + 1;
 Y:;}
```

```
✻ if ( i == n ) goto X;
  i = i + 1;
```

Three iterations

```
     unsigned int j = 0;
     j = j + 1;
☞    j = j + 1
     j = j + 1;
     if ( j == n ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

```
* if ( i == n ) goto X;
  i = i + 1;
```

Three iterations

```
    unsigned int j = 0;
    j = j + 1;
☞   j = j + 1;
    j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

Four iterations

```
    unsigned int j = 0;
    j = j + 1;
☞   j = j + 1;
    j = j + 1
    j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

Exiting after three iterations

```
☞   j = j + 1;
    j = j + 1;
Y:;}
```

```
*  if ( i == n ) goto X;
   i = i + 1;
```

Three iterations

```
     unsigned int j = 0;
     j = j + 1;
☞    j = j + 1;
     j = j + 1;
     if ( j == n ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

Four iterations

```
     unsigned int j = 0;
     j = j + 1;
☞    j = j + 1;
     j = j + 1
     j = j + 1;
     if ( j == n ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
Y:;}
```

✻
```
if ( i == n ) goto X;
i = i + 1;
```

Three iterations

```
    unsigned int j = 0;
    j = j + 1;
    j = j + 1;
☞   j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

Four iterations

```
    unsigned int j = 0;
    j = j + 1;
    j = j + 1;
☞   j = j + 1
    j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

Two iterations

```
*  if ( i == n ) goto X;
   i = i + 1;
```

Three iterations

```
    unsigned int j = 0;
    j = j + 1;
☞  j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

```
    unsigned int j = 0;
    j = j + 1;
☞  j = j + 1
    j = j + 1;
    if ( j == n ) goto Y;
    j = j + 1;
    while ( j != n )   j = j + 1;
Y:;}
```

Two iterations

```
✻ if ( i == n ) goto X;
  i = i + 1;
```

Three iterations

```
     unsigned int j = 0;
     j = j + 1;
☞    j = j + 1;
     if ( j == n ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
  Y:;

   }
```

```
     unsigned int j = 0;
     j = j + 1;
☞    j = j + 1
    j = j + 1;
     if ( j == n ) goto Y;
     j = j + 1;
     while ( j != n )   j = j + 1;
  Y:;
   }
```

Two iterations

*
```
if ( i == n ) goto X;
i = i + 1;
...
j = j + 1
```

Three iterations

```
  unsigned int j = 0;
  j = j + 1;
  j = j + 1;
  if ( j == n ) goto Y;
  j = j + 1;
  while ( j != n )   j = j + 1;
Y:;
☞ }
```

```
  unsigned int j = 0;
  j = j + 1;
  j = j + 1
 j = j + 1;
  if ( j == n ) goto Y;
  j = j + 1;
  while ( j != n )   j = j + 1;
Y:;
☞ }
```

```
void f( unsigned int n )
  {
   posit
     {
      unsigned int i = 0;
      while ( i != n )
         i = i + 1;
     }

   claim
     {
      bool even = true, odd = false;

      for ( unsigned int j = 0; j != n; ++j )
         swap( even, odd );
     }
   ...
```

- Posit mathematical facts before relying on them.
- Mention equalities before they become important.
- Bound loops before entering them.
- Unroll inductions before complex loops.

```
void f( unsigned n )
  {
   posit
     {
      for ( unsigned i = 0; i != n; i = i + 1 )
        {}
     }

   claim
     {
      ...
     }

   for ( unsigned r = n; r != 0; r = r - 1 )
     {}
  }
```

```
void f( unsigned n )
  {
   posit
     {
      for ( unsigned i = 0; i != n; i = i + 1 )   {}
     }

   claim
     {
      unsigned s = n;
      for ( unsigned j = 0; j != n; j = j + 1 )
        {
         posit predecessor( s ) == s - 1;
         s = s - 1;
        }
     }

   for ( unsigned r = n; r != 0; r = r - 1 )   {}
  }
```

```
inline unsigned predecessor( unsigned s )
  {
   unsigned result = 0;
   unsigned next = result + 1;

   while ( next != s )
     {
      result = next;
      next = result + 1;
     }

   return result;
  }
```

```
inline void
counting_is_reversible( const unsigned n )
  {
   claim
     {
      for ( unsigned i = 0; i != n; i = i + 1 )
         {}
     }

   claim implementation;

   claim
     {
      for ( unsigned r = n; r != 0; r = r - 1 )
         {}
     }
  }
```

```
inline void up_down( unsigned i )
  {
   posit i + 1 - 1 == i;
  }


void implementation
counting_is_reversible( const unsigned n )
  {
   unsigned s = n;
   for ( unsigned j = 0; j != n; j = j + 1 )
     {
      up_down( predecessor( s ) );
      s = s - 1;
     }
  }
```

```
template < class ForwardIterator >
inline void reachable( ForwardIterator b, ForwardIterator e )
  {
   while ( b != e )
       ++b;
  }

template < class BidirectionalIterator >
inline void reverse_reachable( BidirectionalIterator b, BidirectionalIterator e )
  {
   while ( e != b )
       --e;
  }
```

```
template < class BidirectionalIterator >
inline void reachability_is_reversible( const BidirectionalIterator b,
                                        const BidirectionalIterator e )
  {
   claim reachable( b, e );
   claim implementation;
   claim reverse_reachable( b, e );
  }

template < class BidirectionalIterator >
inline void implementation reachability_is_reversible( const BidirectionalIterator b,
                                                       const BidirectionalIterator e )
  {
   BidirectionalIterator s = e;
   for ( BidirectionalIterator i = b; i != e; ++i )
     {
      up_down( predecessor_of_end( b, s ) );
      --s;
     }
  }
```

```
template < class ForwardIterator >
inline void predecessor_of_end( ForwardIterator b, ForwardIterator e )
  {
   ForwardIterator next = b;
   ForwardIterator result = next;
   ++next;
   while ( next != e )
     {
      result = next;
      ++next;
     }
   return result;
  }
```

```
template < class BidirectionalIterator >
inline void up_down( const BidirectionalIterator i )
   {
    BidirectionalIterator j = i;
    ++j;
    --j;
    claim implementation;
    claim j == i;
   }

template < class BidirectionalIterator >
void implementation up_down( const BidirectionalIterator i )
   {
    // This is a customization point:
    // specialize this if the proof of up_down is not trivial.
   }
```

Questions?