## Template Functions and Class Templates

© 2010 David A. Smallberg

```
int minimum(int a, int b)
   if (a < b)
     return a;
     return b;
double minimum(double a, double b)
   if (a < b)
      return a;
   else
     return b;
int main()
   cin >> k;
   cout << minimum(k, 10) / 2;</pre>
   double x;
   double y = 3 * minimum(x*x, x+10);
   int z = minimum(0, 3*k - k + 4);
                                    © 2010 David A. Smallberg
```

```
T minimum(T a, T b)
{
   if (a < b)
    return a;
   else
   return b;
}
```

```
template<typename T>
T minimum(T a, T b)
{
    if (a < b)
        return a;
    else
        return b;
}
int main()
{
    int k;
    cin >> k;
    cout << minimum(k, 10) / 2;
    double x;
    ...
    double z = 3 * minimum(x*x, x+10);
    ...
    int z = minimum(0, 3*k - k + 4);
    ...
}</pre>
```

```
template<typename T>
                                                 int minimum(int a, int b)
T minimum(T a, T b)
                                                    if (a < b)
{
                                                       return a;
   if (a < b)
     return a;
                                                       return b;
   else
      return b;
}
int main()
{
   cin >> k;
   cout << minimum(k, 10) / 2;</pre>
   double x;
   double z = 3 * minimum(x*x, x+10);
   int z = minimum(0, 3*k - k + 4);
}
                                      © 2010 David A. Smallberg
```

```
int minimum(int a, int b)
template<typename T>
T minimum(T a, T b)
                                                     if (a < b)
                                                        return a;
   if (a < b)
                                                      else
      return a;
                                                        return b;
   else
                                                  }
      return b;
}
                                                  double minimum(double a, double b)
int main()
                                                     if (a < b)
                                                        return a;
  int k;
cin >> k;
cout << minimum(k, 10) / 2;</pre>
                                                     else
                                                        return b;
   double x;
   double z = 3 * minimum(x*x, x+10);
   int z = minimum(0, 3*k - k + 4);
}
                                       © 2010 David A. Smallberg
```

```
template<typename T>
                                                  int minimum(int a, int b)
T minimum(T a, T b)
                                                      if (a < b)
{
                                                        return a;
   if (a < b)
     return a;
                                                        return b;
   else
     return b;
}
                                                  double minimum(double a, double b)
int main()
                                                     if (a < b)
{
                                                        return a;
  int k;
cin >> k;
cout << minimum(k, 10) / 2;</pre>
                                                     else
                                                       return b;
   double x;
   double z = 3 * minimum(x*x, x+10);
   int z = minimum(0, 3*k - k + 4);
}
                                       © 2010 David A. Smallberg
```

```
template<typename T>
T minimum(T a, T b)
{
    if (a < b)
        return a;
    else
        return b;
}
int main()
{
    int k;
    ... minimum(k, 3.5) ... // Error!
}</pre>
```

```
template<typename T>
                                                string minimum(string a, string b)
T minimum(T a, T b)
                                                    if (a < b)
{
                                                      return a;
   if (a < b)
     return a;
                                                      return b;
   else
     return b;
}
int main()
{
  string s1, s2;
  string s3 = minimum(s1, s2);
}
                                     © 2010 David A. Smallberg
```

```
char* minimum(char* a, char* b)
template<typename T>
T minimum(T a, T b)
                                                 if (a < b) // what does this do?
  return a;
else</pre>
   if (a < b)
     return a;
                                                    return b;
  else
                                              }
     return b;
}
int main()
  char ca2[100];
  cin.getline(ca1, 100);
  cin.getline(ca2, 100);
   char* ca3 = minimum(ca1, ca2);
}
                                    © 2010 David A. Smallberg
```

```
template<typename T>
T minimum(T a, T b)
{
   if (a < b)
     return a;
   else
     return b;
}
char* minimum(char* a, char* b)
   if (strcmp(a,b) < 0)
     return a;
   else
     return b;
int main()
   char ca1[100]; // C string
  char ca2[100];
  cin.getline(ca1, 100);
  cin.getline(ca2, 100);
  char* ca3 = minimum(ca1, ca2);
}
                                     © 2010 David A. Smallberg
```

```
template<typename T>
T minimum(T a, T b)
{
   if (a < b)
        return a;
        else
        return b;
}

int main()
{
   Chicken c1, c2;
   ...
   Chicken c3 = minimum(c1, c2);
   ...
}</pre>
Chicken minimum(chicken a, Chicken b)
{
   if (a < b) // what does this do?
        return a;
        else
        return b;
}

int main()
{
   Chicken c1, c2;
   ...
   Chicken c3 = minimum(c1, c2);
   ...
}

© 2010 David A. Smallberg
```

```
template<typename T>
                                                ExpensiveToCopyThing minimum(
T minimum(T a, T b)
                                                   ExpensiveToCopyThing a,
                                                   ExpensiveToCopyThing b)
{
   if (a < b)
                                                   if (a < b)
     return a;
   else
                                                      return a;
     return b;
                                                   else
                                                      return b;
}
int main()
                                                int minimum(int a, int b)
{
   ExpensiveToCopyThing x, y;
                                                   if (a < b)
                                                      return a;
  ... minimum(x, y) ...;
                                                   else
                                                      return b;
  int m, n;
   ... minimum(m, n) ...;
   . . .
}
                                     © 2010 David A. Smallberg
```

```
ExpensiveToCopyThing minimum(
template<typename T>
T minimum(const T& a, const T& b)
                                                   const ExpensiveToCopyThing& a,
                                                   const ExpensiveToCopyThing& b)
   if (a < b)
                                                   if (a < b)
     return a;
   else
                                                   return a;
else
      return b;
                                                      return b;
}
                                                }
int main()
                                                int minimum(const int& a, const int& b)
   ExpensiveToCopyThing x, y;
                                                   if (a < b)
                                                      return a;
   ... minimum(x, y) ...;
                                                   else
   . . .
                                                      return b;
  int m, n;
                                                }
  ... minimum(m, n) ...;
}
                                     © 2010 David A. Smallberg
```

```
template<typename T>
                                                double sum(const double a[], int n)
T sum(const T a[], int n)
                                                   double total = 0;
                                                  for (int k = 0; k < n; k++)
  T total = 0;
  for (int k = 0; k < n; k++)
                                                    total += a[k];
    total += a[k];
                                                  return total;
  return total;
}
int main()
  double da[100];
  cout << sum(da, 10);</pre>
}
                                     © 2010 David A. Smallberg
```

```
double sum(const double a[], int n)
template<typename T>
T sum(const T a[], int n)
                                                 double total = 0;
  T total = 0;
                                                 for (int k = 0; k < n; k++)
  for (int k = 0; k < n; k++)
                                                   total += a[k];
    total += a[k];
                                                 return total;
  return total;
}
                                              string sum(const string a[], int n)
int main()
                                                 string total = 0; // uh-oh
  double da[100];
                                                for (int k = 0; k < n; k++)
                                                  total += a[k];
  cout << sum(da, 10);</pre>
                                                 return total;
                                              }
   string sa[10] = {
    "This ", "is ", "a ", "test."
  };
  string s = sum(sa, 4);
}
                                    © 2010 David A. Smallberg
```

```
template<typename T>
                                               double sum(const double a[], int n)
T sum(const T a[], int n)
                                                  double total = double();
                                                  for (int k = 0; k < n; k++)
  T total = T();
                                                    total += a[k];
  for (int k = 0; k < n; k++)
     total += a[k];
                                                  return total;
  return total;
                                               }
}
                                               string sum(const string a[], int n)
int main()
{
                                                  string total = string();
  double da[100];
                                                  for (int k = 0; k < n; k++)
                                                    total += a[k];
   cout << sum(da, 10);</pre>
                                                  return total;
   string sa[10] = {
     "This ", "is ", "a ", "test."
  }:
   string s = sum(sa, 4);
}
                                    © 2010 David A. Smallberg
```

```
class StackOfInt
                                               void StackOfInt::pop()
                                               {
  public:
                                                  m_top--; // trouble later if was empty
  StackOfInt();
                                               }
  void push(int x);
  void pop();
                                               int StackOfInt::top() const
  int top() const;
                                               ł
  int size() const;
                                                  return m_data[m_top-1];
                                                         // undefined if empty
 private:
  int m_data[100];
  int m_top;
                                               int StackOfInt::size() const
StackOfInt::StackOfInt()
                                                  return m_top;
: m_top(0)
{}
void StackOfInt::push(int x)
{
  m_{data}[m_{top}] = x; // undefined if full
  m_top++;
                                    © 2010 David A. Smallberg
```

```
class StackOfInt
                                               void StackOfInt::pop()
                                                {
  public:
                                                  m_top--; // trouble later if was empty
   StackOfInt();
  void push(int x);
  void pop();
                                               int StackOfInt::top() const
  int top() const;
  int size() const;
                                                  return m_data[m_top-1];
                                                           // undefined if empty
  private:
  int m_data[100];
  int m_top;
                                               int StackOfInt::size() const
};
StackOfInt::StackOfInt()
                                                  return m_top;
: m_top(0)
void StackOfInt::push(int x)
   m_{data}[m_{top}] = x; // undefined if full
   m_top++;
}
                                     © 2010 David A. Smallberg
```

```
template<typename T>
                                               template<typename T>
class Stack
                                               void Stack<T>::pop()
{
                                               {
  public:
                                                 m_top--; // trouble later if was empty
  Stack();
  void push(const T& x);
  void pop();
                                               template<typename T>
  T top() const;
                                               T Stack<T>::top() const
  int size() const;
                                               {
  private:
                                                  return m_data[m_top-1];
                                                          // undefined if empty
  T m_data[100];
  int m_top;
};
                                               template<typename T>
template<typename T>
                                               int Stack<T>::size() const
Stack<T>::Stack()
: m_top(0)
                                                  return m_top;
                                               }
{}
template<typename T>
void Stack<T>::push(const T& x)
{
  m_data[m_top] = x; // undefined if full
  m_top++;
}
                                    © 2010 David A. Smallberg
```

```
template<typename T>
                                               Stack<int>::Stack()
class Stack
                                               : m_top(0)
                                               {}
  public:
   Stack();
                                               void Stack<int>::push(const int& x)
   void push(const T& x);
                                               {
                                                 m_data[m_top] = x; // undefined if full
  private:
                                                 m_top++;
  T m_data[100];
   int m_top;
};
                                               Stack<int>::~Stack()
class Coord
                                               {}
  public:
                                               Stack<Coord>::Stack()
   Coord(int r, int c);
                                               : m_top(0)
   Coord();
                                               {}
...
};
                                               void Stack<Coord>::push(const Coord& x)
int main()
                                                 m_data[m_top] = x; // undefined if full
   Stack<int> si;
                                                 m_top++;
   si.push(3);
                                               }
   Stack<Coord> sc;
   sc.push(Coord(3,5));
                                               Stack<Coord>::~Stack()
                                               {}
                                    © 2010 David A. Smallberg
```