

Template Functions and Class Templates

© 2018 David A. Smallberg

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

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

int main()
{
    int k;
    cin >> k;
    cout << minimum(k, 10) / 2;
    double x;
    ...
    double y = 3 * minimum(x*x, x+10);
    ...
    int z = minimum(0, 3*k - k + 4);
    ...
}
```

© 2018 David A. Smallberg

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

© 2018 David A. Smallberg

```
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);
    ...
}
```

© 2018 David A. Smallberg

```

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);
    ...
}

```

© 2018 David A. Smallberg

```

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);
    ...
}

```

© 2018 David A. Smallberg

```

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);
    ...
}

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

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

```

© 2018 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!
}

```

© 2018 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);
    ...
}

```

```

Chicken minimum(Chicken a, Chicken b)
{
    if (a < b) // what does this do?
        return a;
    else
        return b;
}

```

© 2018 David A. Smallberg

```

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

int main()
{
    string s1, s2;
    ...
    string s3 = minimum(s1, s2);
    ...
}

```

```

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

```

© 2018 David A. Smallberg

```

template<typename T>
T minimum(T a, T b)
{
    if (a < b)
        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);
    ...
}

```

```

char* minimum(char* a, char* b)
{
    if (a < b) // what does this do?
        return a;
    else
        return b;
}

```

© 2018 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);
    ...
}

```

© 2018 David A. Smallberg

```

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

int main()
{
    ExpensiveToCopyThing x, y;
    ...
    ... minimum(x, y) ...;
    ...
    int m, n;
    ...
    ... minimum(m, n) ...;
    ...
}

```

```

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

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

```

© 2018 David A. Smallberg

```

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

int main()
{
    ExpensiveToCopyThing x, y;
    ...
    ... minimum(x, y) ...;
    ...
    int m, n;
    ...
    ... minimum(m, n) ...;
    ...
}

```

```

ExpensiveToCopyThing minimum(
    const ExpensiveToCopyThing& a,
    const ExpensiveToCopyThing& b)
{
    if (a < b)
        return a;
    else
        return b;
}

int minimum(const int& a, const int& b)
{
    if (a < b)
        return a;
    else
        return b;
}

```

© 2018 David A. Smallberg

```

template<typename T>
T sum(const T a[], int n)
{
    T total = 0;
    for (int k = 0; k < n; k++)
        total += a[k];
    return total;
}

int main()
{
    double da[100];
    ...
    cout << sum(da, 10);
    ...
}

double sum(const double a[], int n)
{
    double total = 0;
    for (int k = 0; k < n; k++)
        total += a[k];
    return total;
}

```

© 2018 David A. Smallberg

```

template<typename T>
T sum(const T a[], int n)
{
    T total = 0;
    for (int k = 0; k < n; k++)
        total += a[k];
    return total;
}

int main()
{
    double da[100];
    ...
    cout << sum(da, 10);
    ...
    string sa[10] = {
        "This ", "is ", "a ", "test."
    };
    string s = sum(sa, 4);
    ...
}

double sum(const double a[], int n)
{
    double total = 0;
    for (int k = 0; k < n; k++)
        total += a[k];
    return total;
}

string sum(const string a[], int n)
{
    string total = 0; // uh-oh
    for (int k = 0; k < n; k++)
        total += a[k];
    return total;
}

```

© 2018 David A. Smallberg


```

template<typename T>
T sum(const T a[], int n)
{
    T total = T();
    for (int k = 0; k < n; k++)
        total += a[k];
    return total;
}

int main()
{
    double da[100];
    ...
    cout << sum(da, 10);
    ...
    string sa[10] = {
        "This ", "is ", "a ", "test."
    };
    string s = sum(sa, 4);
    ...
}

double sum(const double a[], int n)
{
    double total = double();
    for (int k = 0; k < n; k++)
        total += a[k];
    return total;
}

string sum(const string a[], int n)
{
    string total = string();
    for (int k = 0; k < n; k++)
        total += a[k];
    return total;
}

```

© 2018 David A. Smallberg

```

class StackOfInt
{
public:
    StackOfInt();
    void push(int x);
    void pop();
    int top() const;
    int size() const;
private:
    int m_data[100];
    int m_top;
};

StackOfInt::StackOfInt()
: m_top(0)
{}

void StackOfInt::push(int x)
{
    m_data[m_top] = x; // undefined if full
    m_top++;
}

void StackOfInt::pop()
{
    m_top--; // trouble later if was empty
}

int StackOfInt::top() const
{
    return m_data[m_top-1];
    // undefined if empty
}

int StackOfInt::size() const
{
    return m_top;
}

```

© 2018 David A. Smallberg

```

class StackOfInt
{
public:
    StackOfInt();
    void push(int x);
    void pop();
    int top() const;
    int size() const;
private:
    int m_data[100];
    int m_top;
};

StackOfInt::StackOfInt()
: m_top(0)
{}

void StackOfInt::push(int x)
{
    m_data[m_top] = x; // undefined if full
    m_top++;
}

void StackOfInt::pop()
{
    m_top--; // trouble later if was empty
}

int StackOfInt::top() const
{
    return m_data[m_top-1];
    // undefined if empty
}

int StackOfInt::size() const
{
    return m_top;
}

```

© 2018 David A. Smallberg

```

template<typename T>
class Stack
{
public:
    Stack();
    void push(const T& x);
    void pop();
    T top() const;
    int size() const;
private:
    T m_data[100];
    int m_top;
};

template<typename T>
Stack<T>::Stack()
: m_top(0)
{}

template<typename T>
void Stack<T>::push(const T& x)
{
    m_data[m_top] = x; // undefined if full
    m_top++;
}

template<typename T>
void Stack<T>::pop()
{
    m_top--; // trouble later if was empty
}

template<typename T>
T Stack<T>::top() const
{
    return m_data[m_top-1];
    // undefined if empty
}

template<typename T>
int Stack<T>::size() const
{
    return m_top;
}

```

© 2018 David A. Smallberg

```

template<typename T>
class Stack
{
public:
    Stack();
    void push(const T& x);
    ...
private:
    T m_data[100];
    int m_top;
};

class Coord
{
public:
    Coord(int r, int c);
    Coord();
    ...
};

int main()
{
    Stack<int> si;
    si.push(3);
    Stack<Coord> sc;
    sc.push(Coord(3,5));
}

Stack<int>::Stack()
: m_top(0)
{}

void Stack<int>::push(const int& x)
{
    m_data[m_top] = x; // undefined if full
    m_top++;
}

Stack<int>::~Stack()
{}

Stack<Coord>::Stack()
: m_top(0)
{}

void Stack<Coord>::push(const Coord& x)
{
    m_data[m_top] = x; // undefined if full
    m_top++;
}

Stack<Coord>::~Stack()
{}

```

© 2018 David A. Smallberg