Variadic Templates Solutions

Variadic function templates

- What is a variadic function template?
  - A variadic function template is a function template which takes a variable number of arguments

Parameter Packs

- What is meant by the following terms?
  - Template parameter pack
  - A template parameter pack is a list of the types of the arguments
  - Function parameter pack
  - A function parameter pack is a list of the arguments
- Write down a declaration of a variadic function template

```
template <typename... Args>      // Args is a list of types (template parameter pack)
void func(Args... args);         // args is a list of arguments (function parameter pack)
```

Type deduction

- Given the following declaration of func and definitions of i, d and s

```
template <typename... Args>
void func(Args... args);

int i{42};
double d{0.0};
string s{"text"};
```

- Write down the signature of the functions which would be instantiated by the following calls:

```
func(s);          // Instantiated as func(string);
func(i, d, s);    // Instantiated as func(int, double, string);
```

sizeof... operator

- What is the purpose of the sizeof... operator?
  - When passed a parameter pack, the sizeof... operator returns the number of elements in the pack
  - This function is available at compile time
- Write a simple program that demonstrates the use of the sizeof... operator

Pack expansion

- What is a "pack expansion"?
  - In a context where a list of arguments is expected, if we put the function parameter pack followed by ..., the compiler will replace it with the list of arguments

Pack processing

- Describe a typical way of processing a function parameter pack
  - Typically, recursion is used to process a function pack
  - The function pack is passed as two arguments
  - The first argument is the "front" of the list of arguments
  - The second argument is the "rest" of the list of arguments
  - The front element is processed
  - The function is called again, passing the "rest" as its argument
  - The next element in the list becomes the new "front" element
  - The recursion terminates with a function that takes a single argument, representing the last element in the list

Pack processing

- Write a variadic function template which processes its function parameter pack
- As each element in the parameter pack is processed, print it out together with the number of elements which remain to be processed