Lab 3

Function Overloading

Introduction

The goal of these exercises is to investigate function overloading and default arguments. The first exercise involves answering several questions about finding which overloaded function matches a function invocation. The second exercise asks you to implement a set of overloaded **Trace** functions that can be used for simple tracing in a C++ program, encapsulating the actual mechanism that is used to output the data.

Suggested Time: 15 minutes.

Exercise 1. Answer the following questions:

(a) Which one of these functions:

```
void foo(void *)
void foo(const char*)
void foo(char)
void foo(int)
```

is matched by the function call:

```
foo("123")
```

(b) Which one of these functions:

```
void foo(double)
void foo(float)
void foo(char *)
void foo(int)
```

is matched by the function call:

```
foo(2.71828)
```

(c) Which one of these functions:

```
            void foo(const char *)
            void foo(float)
            void foo(int *)
```

is matched by the function call:

```
            foo("0xff45b2c2")
```

Exercise 2. Overloaded Trace Functions

Create a header file **Trace.h** that implements a set of three overloaded functions **Trace** that will write various standard data types to an output device, followed by a **newline** character.

The first function takes a single **const char *** parameter, the second takes two **const char *** parameters, and the third takes a **const char *** parameter and an **int** parameter. The calling program will use the same interface whatever actual output mechanism is used in implementing Trace.

This encapsulation will facilitate porting programs to other environments, such as a GUI. In your implementation use standard C++ stream I/O with your output going to **cout**. Use the supplied test program **TstTrace.cpp** to test your functions.

Lab 3   Answers

Exercise 1.

(a) **void foo(void *).** A **void** pointer will match any pointer. If the function **void foo(char *)** were provided, that would be the match instead, because it would be an exact match.

(b) **void foo(double).** The float constant 2.71828 is promoted to a **double**, which is a "widening" that does not lose information. A widening takes precedence over the "narrowing" conversion to **int**, which loses information.

(c) The match is ambiguous, because both **void foo(char**) and **void foo(float)** match by conversion without loss of information. The function void **foo(int),** if provided, would be an exact match.