Function Pointers

Gabriel Hugh Elkaim
Winter 2014

Function Pointers

• Pointers may also be used to point to functions
• Provides a more flexible way to call a function, by providing a choice of which function to call
• Makes it possible to pass functions to other functions
• Not extremely common, but very useful in the right situations

Function Pointers

• A function pointer is declared much like a function prototype:

\[
\text{int } (*fp)(\text{int } x);
\]

• Here, we have declared a function pointer with the name \( fp \)
  – The function it points to must take one int parameter
  – The function it points to must return an int

Function Pointers

• A function pointer is initialized by setting the pointer name equal to the function name

If we declare the following:

\[
\text{int } (*fp)(\text{int } x);
\]

\[
\text{int } \text{foo}(\text{int } x);
\]

We can initialize the function pointer like this:

\[
fp = \text{foo}; \quad \text{//fp now points to foo}
\]
### Function Pointers

**Calling a Function via a Function Pointer**

- The function pointed to by `fp` from the previous slide may be called like this:

\[
y = fp(x);
\]

- This is the same as calling the function directly:

\[
y = foo(x);
\]

### Function Pointers

**Passing a Function to a Function**

#### Example 1: Understanding the Mechanism

```c
int x;
int foo(int a, int b); // Function prototype
int bar(int a, int b); // Function prototype

// Function definition with function pointer parameter
int foobar(int a, int b, int (*fp)(int, int))
{
    return fp(a, b); // Call function passed by pointer
}

void main(void)
{
    x = foobar(5, 12, &foo); // Pass address of foo
}
```

#### Example 2: Evaluate a Definite Integral (approximation)

```c
float integral(float a, float b, float (*f)(float))
{
    float sum = 0.0;
    float x;
    int n;

    // Evaluate integral \( \int_a^b f(x) \) dx
    for (n = 0; n <= 100; n++)
    {
        x = ((n / 100.0) * (b - a)) + a;
        sum += f(x) * (b - a) / 101.0;
    }
    return sum;
}
```

Adapted from example at: http://en.wikipedia.org/wiki/Function_pointer

### Lab Exercise 13

**Function Pointers**
Lab 13
Function Pointers

• Open the lab Project:
  On the class website /Examples/Lab13.zip -> Load “Lab13.X”

  1 Open MPLAB®X and select Open Project Icon (Ctrl + Shift + O)
  Open the Project listed above.
  If you already have a project open in MPLAB X, close it by “right clicking” on the open project and selecting “Close”

Lab 13
Function Pointers

• Compile and run the code:
  2 Click on the Debug Project button.
  3 If no errors are reported, click on Continue button to start the program.
  4 Click on the Pause button.

Lab 13
Function Pointers

• Results

  Three separate functions are integrated over the interval 0 to 1:

  \[
  y_1 = \int_0^1 x \, dx = \frac{1}{2} x^2 + C \mid_0^1 = 0.500000
  \]

  \[
  y_2 = \int_0^1 x^2 \, dx = \frac{1}{3} x^3 + C \mid_0^1 = 0.335000
  \]

  \[
  y_3 = \int_0^1 x^3 \, dx = \frac{1}{4} x^4 + C \mid_0^1 = 0.252500
  \]

Exercise 13
Function Pointers

Function to Evaluate: xsquared()

```c
//xsquared() for Exercise 13
float xsquared(float x)
{
    return (x * x);
}
```
Exercise 13
Function Pointers

/*============================================================================
FUNCTION:     integral() DESCRIPTION:  Evaluates the integral of the function passed to it over the
interval a to b.
PARAMETERS:   interval end points a & b and function to integrate
RETURNS:      integral of function f over interval a to b
REQUIREMENTS: none
SOURCE:       Adapted from example at:
http://en.wikipedia.org/wiki/Function_pointer
============================================================================*/

float integral(float a, float b, float (*f)(float))
{
    float sum = 0.0;
    float x;
    int n;
    //Evaluate integral[a,b] f(x) dx
    for (n = 0; n <= 100; n++) {
        x = ((n / 100.0) * (b-a)) + a;
        sum += (f(x) * (b-a)) / 101.0;
    }
    return sum;
}

Exercise 13
Conclusions

- Function pointers, while not frequently used, can provide a very convenient mechanism for
  passing a function to another function
- Many other possible applications exist
  - Jump tables
  - Accommodating multiple calling conventions
  - Callback functions (used in Windows™)
  - Call different versions of a function under different circumstances

Questions?