Functions and Pointers

Last updated 5/16/24

These slides introduce pointers in functions

Functions and Pointers

- Function Input and Output
  - Input – through actual parameters
  - Output – through return value
    - Only one value can be returned
  - User Input/Output – through side effects
    - printf
    - scanf

```
int main(void){
  float checking;
  float savings;
  float int_rate;
   ...
  checking = update_acct(checking, int_rate);
  savings = update_acct(savings, int_rate);
  return 0;
}

float update_acct(float bal, float ir){
  bal += bal * ir;
  return bal;
}
```

Functions and Pointers

- Pointers and functions
  - Pointers allow us to use called functions to change values in the calling function
  - Instead of passing variables in the parameter list (remember copies are made and then relinquished) we can pass pointers
  - Pointers allow us to modify the calling programs variables by memory reference

Functions and Pointers

- Function Declaration
  - Indicate that a pointer is being passed in the Formal Parameter List

```
void update_acct(float * balance_ptr,   float int_rate);
```

passing a pointer of type float

passing a float

Functions and Pointers

- Function Definition
  - Indicate that a pointer is being passed in the Formal Parameter List
  - Operate on the variables pointed to by the pointers via the dereference operator

```
void update_acct(float * balance_ptr, float int_rate){
    *balance_ptr = *balance_ptr  + *balance_ptr * int_rate;
    return;
}
```

the value pointed to by balance_ptr is assigned the value of the value pointed to by balance_ptr + the value pointed to by balance_ptr times int_rate

Functions and Pointers

```
void update_acct(float* balance_ptr, float int_rate){
    *balance_ptr += *balance_ptr * int_rate;
    return;
}
```

- Function Call
  - Pass a pointer variable in the Actual Parameter List

    or
  - Pass the address to the variable in the Actual Parameter List

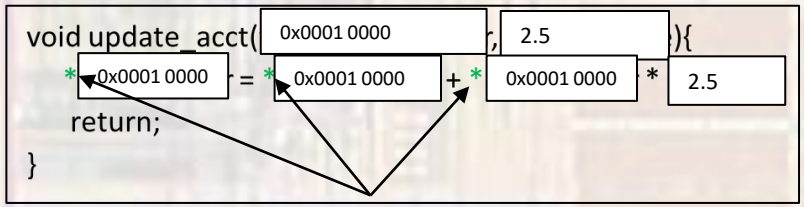
```
int main(void){
  float checking;
  float savings;
  float int_rate;
  float * check_ptr;              // ptr variable to a float variable
  check_ptr = &checking
  …
  update_acct(check_ptr, int_rate);      // using ptr variable
  update_acct(&savings, int_rate);       // using address of a variable
  return 0;
}
```

Functions and Pointers

- Usage
  - Pass a pointer variable in the Actual Parameter List

```
int main(void){
  float checking;                    // stored in 0x0001 0000
  float int_rate;                    // stored in 0x0001 0004
  Int_rate = 2.5;
  checking = 1000;
  float * check_ptr;                 // ptr variable to a float variable
  check_ptr = &checking              // check_ptr has the value 0x0001 0000
  …
  update_acct(check_ptr, int_rate);  // looks like  update_acct(0x0001 0000, 2.5)
  return 0;
}
```

```
void update_acct(float* balance_ptr, float int_rate){
   *balance_ptr = *balance_ptr  + *balance_ptr * int_rate;
    return;
}
```



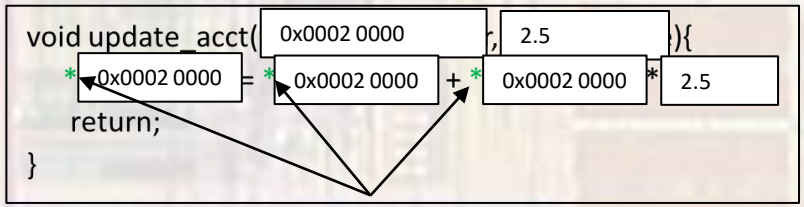
value pointed to by

Functions and Pointers

- Usage
  - Pass the address to the variable in the Actual Parameter List

```
int main(void){
   float savings;                    // stored in 0x0002 0000
   float int_rate;                   // stored in 0x0001 0004
   Int_rate = 2.5;
   savings = 1000;
   …
   update_acct(&savings, int_rate);  // looks like  update_acct(0x0002 0000, 2.5)
   return 0;
}
```

```
void update_acct(float* balance_ptr, float int_rate){
    *balance_ptr = *balance_ptr  + *balance_ptr * int_rate;
     return;
}
```



value pointed to by