CMSC 104 - Lecture 16
Park , adapted by C Grasso

Functions: Part 3 of 4

Functions - Part 3

Topics

- Parameter Passing by Value (review)
- Parameter Passing by Reference

Parameter Passing by Value

- **Formal parameters** are the parameters that appear in the function *header* definition.

```
float AverageTwo (int num1, int num2);
```

- **Actual parameters** are the values that are specified in the function *call*.

```
average = AverageTwo (value1, value2) ;
```

- Actual and formal parameters are matched by *position*. Each formal parameter receives the value of its corresponding actual parameter.

Parameter Passing by Value

- Corresponding actual and formal parameters do not have to have the same name, but they may.
- Corresponding actual and formal parameters must be of the same data type, with some exceptions.

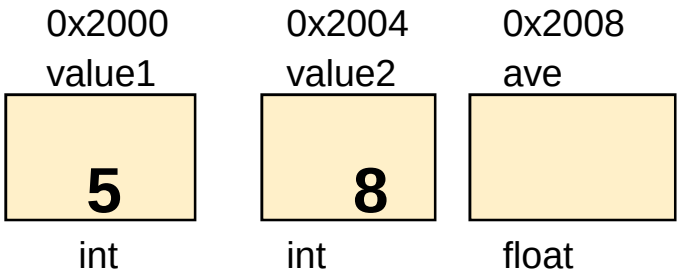
Local Variables

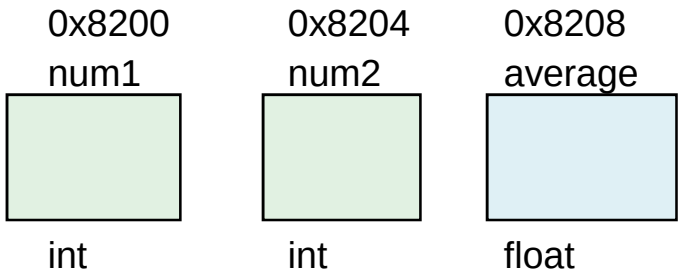
- Functions only "see" (have access to) their own **local variables**.
  - This includes main( ) .
- Formal parameters are declarations of local variables.
  - The values passed are assigned to those variables.
- Other local variables can be declared within the function body.

Parameter Passing and Local Variables

```
#include <stdio.h>
float AverageTwo (int num1, int num2) ;

int main ( )
{
    float ave ;
    int value1 = 5 ;
    int value2 = 8 ;
    ave = AverageTwo (value1, value2)  ;
    return 0 ;
}
```

```
float AverageTwo (int num1, int num2)
{
    float average ;
    average = (num1 + num2) / 2.0 ;
    return   average ;
}
```

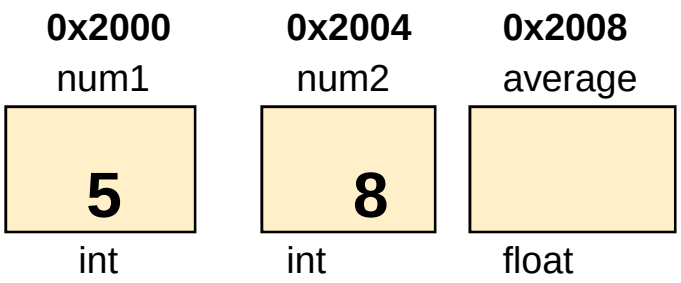


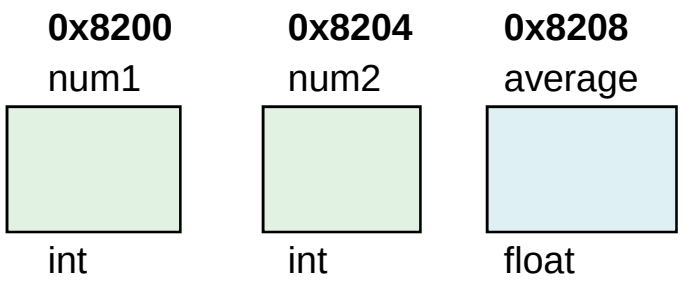


Same Name – Different Memory Locations

```
#include <stdio.h>
float AverageTwo (int num1, int num2) ;

int main ( )
{
    float  ave ;
    int    num1 = 5 ;
    int    num2 = 8 ;
    ave = AverageTwo (num1, num2)  ;
    return 0 ;
}
```

```
float AverageTwo (int num1, int num2)
{
    float average ;
    average = (num1 + num2) / 2.0 ;
    return   average ;
}
```

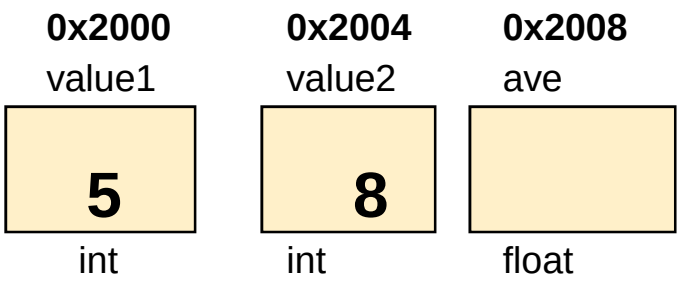


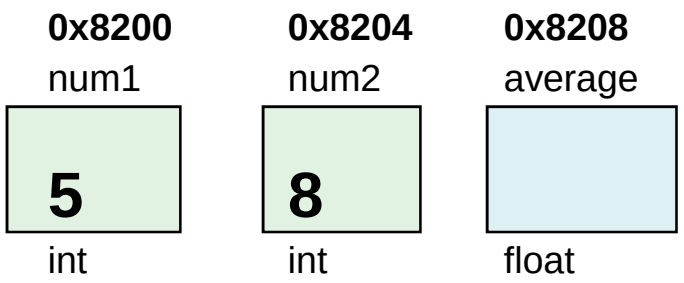


Same Name – Different Memory Locations

```
#include <stdio.h>
float AverageTwo (int num1, int num2) ;

int main ( )
{
    float  ave ;
    int    num1 = 5 ;
    int    num2 = 8 ;
    ave = AverageTwo (num1, num2)  ;
    return 0 ;
}
```

```
float AverageTwo (int num1, int num2)
{
    float average ;
    average = (num1 + num2) / 2.0 ;
    return   average ;
}
```

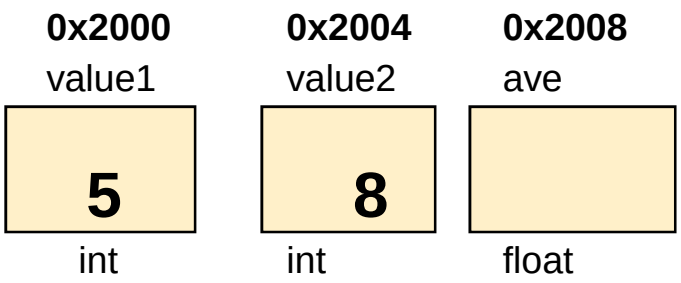


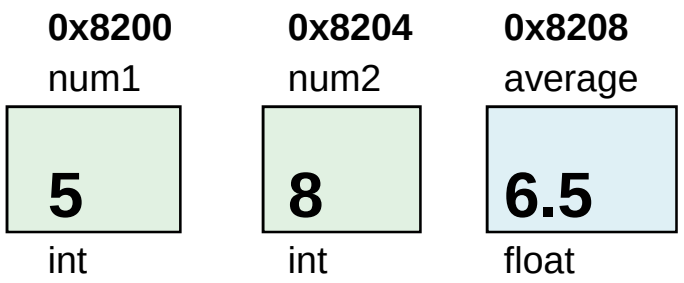


Same Name – Different Memory Locations

```
#include <stdio.h>
float AverageTwo (int num1, int num2) ;

int main ( )
{
    float  ave ;
    int    num1 = 5 ;
    int    num2 = 8 ;
    ave = AverageTwo (num1, num2)  ;
    return 0 ;
}
```

```
float AverageTwo (int num1, int num2)
{
      float average ;
      average = (num1 + num2) / 2.0 ;
      return   average ;
}
```

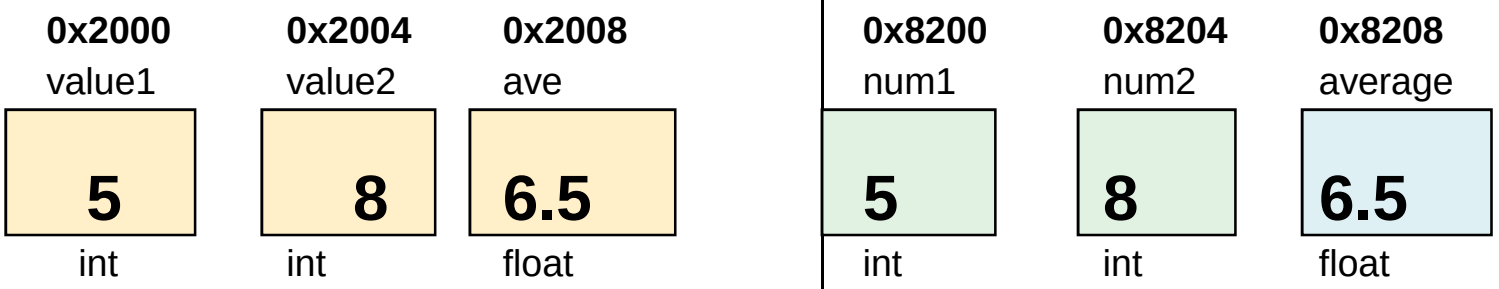




Same Name – Different Memory Locations

```
#include <stdio.h>
float AverageTwo (int num1, int num2) ;

int main ( )
{
    float  ave ;
    int    num1 = 5 ;
    int    num2 = 8 ;
    ave = AverageTwo (num1, num2)  ;
    return 0 ;
}
```

```
float AverageTwo (int num1, int num2)
{
      float average ;
      average = (num1 + num2) / 2.0 ;
      return   average ;
}
```



Changes to Local Variables Do NOT Change Other Variables with the Same Name

```
#include <stdio.h>
void  AddOne (int num1) ;
int main ( )
{
     int num1 = 5 ;

      AddOne (num1) ;
     printf ("In main: ") ;
     printf ("num1 = %d \n", num1) ;

     return  0 ;

}
```

num1

5

```
void    AddOne (int num1)
{
          num1++ ;
          printf ("In AddOne: ") ;
          printf ("num1 = %d\n", num1) ;

}
```

num1

OUTPUT

Changes to Local Variables Do NOT Change Other Variables with the Same Name

```
#include <stdio.h>
void  AddOne (int num1) ;
int main ( )
{
     int num1 = 5 ;

      AddOne (num1) ;
     printf ("In main: ") ;
     printf ("num1 = %d \n", num1) ;

     return  0 ;

}
```

num1

5

```
void    AddOne (int num1)
{
          num1++ ;
          printf ("In AddOne: ") ;
          printf ("num1 = %d\n", num1) ;

}
```

num1

5

OUTPUT

Changes to Local Variables Do NOT Change Other Variables with the Same Name

```
#include <stdio.h>
void  AddOne (int num1) ;
int main ( )
{
     int num1 = 5 ;

     AddOne (num1) ;
     printf ("In main: ") ;
     printf ("num1 = %d \n", num1) ;

     return    0  ;

}
```

num1

5

```
void    AddOne (int num1)
{
          num1++ ;
          printf ("In AddOne: ") ;
          printf ("num1 = %d\n", num1) ;

}
```

num1

6

OUTPUT

```
In AddOne: num1 = 6
```

Changes to Local Variables Do NOT Change Other Variables with the Same Name

```
#include <stdio.h>
void  AddOne (int num1) ;
int main ( )
{
    int num1 = 5 ;

     AddOne (num1) ;
    printf ("In main: ") ;
    printf ("num1 = %d \n", num1) ;

    return  0 ;

}
```

num1

5

```
void    AddOne (int num1)
{
            num1++ ;
            printf ("In AddOne: ") ;
            printf ("num1 = %d\n", num1) ;

}
```

num1

6

OUTPUT

```
In AddOne: num1 = 6
In main:       num1 = 5
```

Parameter Passing by Reference

- **Formal parameters** are the parameters that appear in the function *header* definition.

```
AddOne(int *num1);
```

- **Actual parameters** are the values that are specified in the function *call*.

```
AddOne(&num1) ;
```

- Actual and formal parameters are matched by *position*. Each formal parameter receives the value of its corresponding actual parameter.

Parameter Passing by Reference

- When a formal parameter is passed by **reference**, the **memory address** of the variable is passed.

- The formal parameter reference is indicated by prefixing the variable name with *

```
AddOne(int *num1);
```

- When an **actual parameter** is passed in the function ***call*** the variable is prefixed with &

```
AddOne(&num1) ;
```

Parameter Passing by Reference

- Corresponding actual and formal parameters do not have to have the same name (but they may).

- Corresponding actual and formal parameters must be of the same data type
  - The * becomes part of the data type

```
AddOne(int *num1);
```

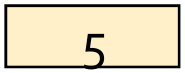
Local Variables

- Variables declared within the function body are called **local variables**;
- Functions only "see" (have access to) their own **local variables**.
  - This includes main( ) .
- However, formal parameters may refer to the memory address of variables in other functions.
  - If a memory addresses of a variable is passed, the **contents** (value) of that location may be changed.

Changes to Reference Variables DO Change the Value in other Variables

```
#include <stdio.h>
void AddOne (int *number) ;

int main ( )
{
    int num1 = 5 ;
    AddOne ( &num1) ;
    printf ("In main: ") ;
    printf ("num1 = %d \n", num1) ;
    return 0 ;
}
```

num1 (0x2008)



int

```
void AddOne (int *num1)
{
        (*num1)++ ;
        printf ("In AddOne: ") ;
        printf ("num1 = %d\n", *num1) ;
}
```

*num1

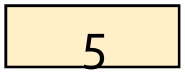
int

OUTPUT

Changes to Reference Variables DO Change the Value in other Variables

```
#include <stdio.h>
void AddOne (int *number) ;

int main ( )
{
    int num1 = 5 ;
    AddOne ( &num1) ;
    printf ("In main: ") ;
    printf ("num1 = %d \n", num1) ;
    return 0 ;
}
```

num1 (0x2008)



int

```
void AddOne (int *num1)
{
        *num1++ ;
        printf ("In AddOne: ") ;
        printf ("num1 = %d\n", *num1) ;
}
```

*num1

int

OUTPUT

Changes to Reference Variables DO Change the Value in other Variables

```
#include <stdio.h>
void AddOne (int *number) ;

int main ( )
{
    int num1 = 5 ;
    AddOne ( &num1) ;
    printf ("In main: ") ;
    printf ("num1 = %d \n", num1) ;
    return 0 ;
}
```

num1 (0x2008)

int

```
void AddOne (int *num1)
{
        *num1++ ;
        printf ("In AddOne: ") ;
        printf ("num1 = %d\n", *num1) ;
}
```

*num1

0x2008

int

OUTPUT

Changes to Reference Variables DO Change the Value in other Variables

```
#include <stdio.h>
void AddOne (int *number) ;

int main ( )
{
    int num1 = 5 ;
    AddOne ( &num1) ;
    printf ("In main: ") ;
    printf ("num1 = %d \n", num1) ;
    return 0 ;
}
```

num1 (0x2008)

6

int

```
void AddOne (int *num1)
{
    *num1++ ;
    printf ("In AddOne: ") ;
    printf ("num1 = %d\n", *num1) ;
}
```

*num1

0x2008

int

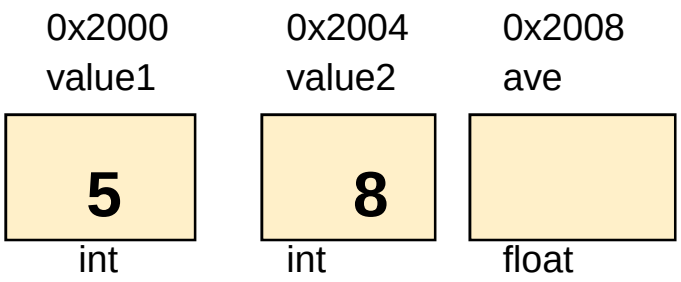
OUTPUT

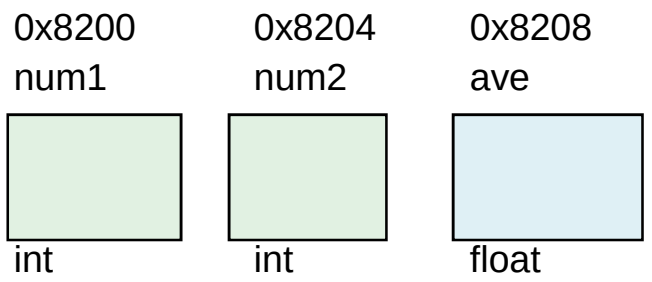
```
In AddOne: *num1 = 6
```

Parameter Passing by Reference and Local Variables

```
#include <stdio.h>
AverageTwo (int num1, int  num2, float *ave) ;

int main ( )
{
    float ave ;
    int value1 = 5, value2 = 8 ;

    AverageTwo (value1, value2, &ave)  ;
    ...
   return 0 ;
}
```

```
AverageTwo (int  num1, int  num2, float *ave)
{


  *ave = (num1 + num2) / 2.0 ;
  return;
}
```

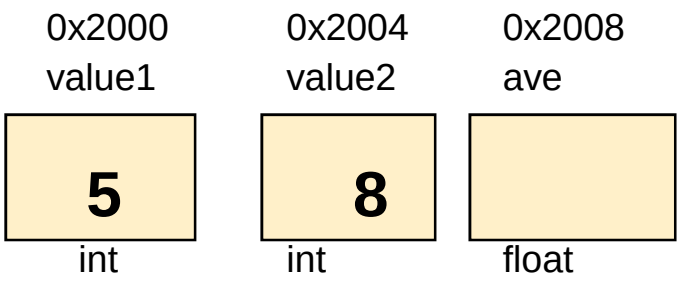


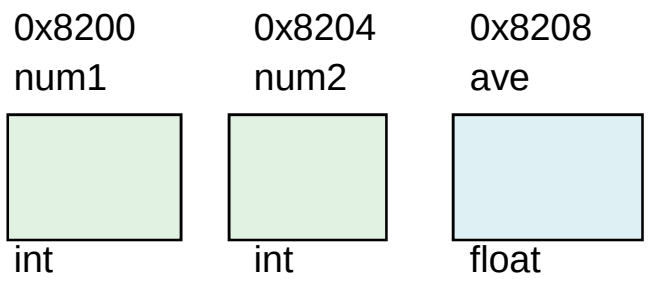


Parameter Passing by Reference and Local Variables

```
#include <stdio.h>
AverageTwo (int num1, int  num2, float *ave) ;

int main ( )
{
     float ave ;
     int value1 = 5, value2 = 8 ;

     AverageTwo (value1, value2, &ave)  ;
     ...
    return 0 ;
}
```

```
AverageTwo (int  num1, int  num2, float *ave)
{


  *ave = (num1 + num2) / 2.0 ;
  return;
}
```

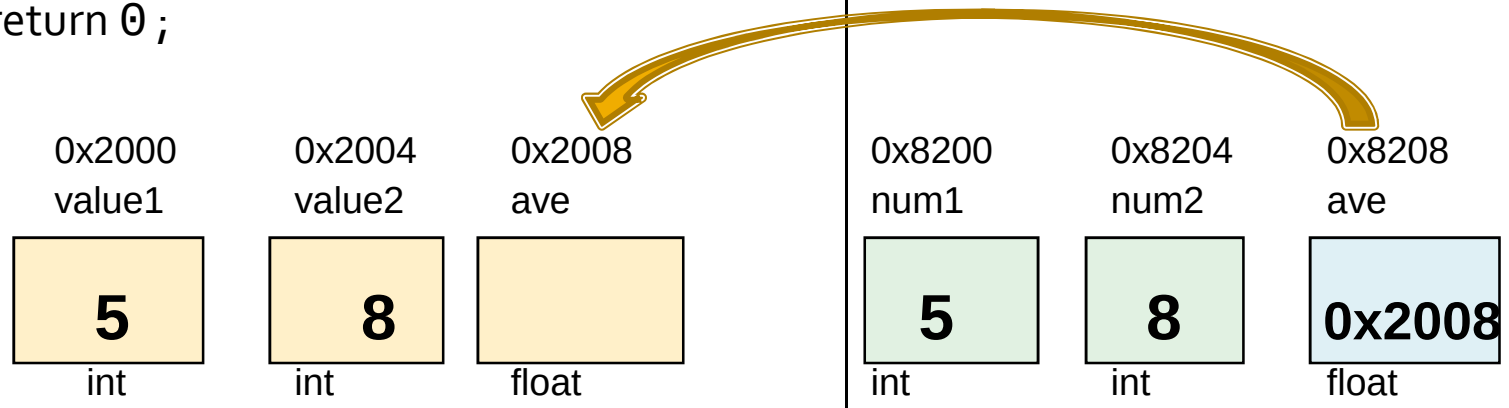




Parameter Passing by Reference and Local Variables

```
#include <stdio.h>
AverageTwo (int num1, int  num2, float *ave) ;

int main ( )
{
     float ave ;
     int value1 = 5, value2 = 8 ;

     AverageTwo (value1, value2, &ave)  ;
     ...
    return 0 ;
}
```

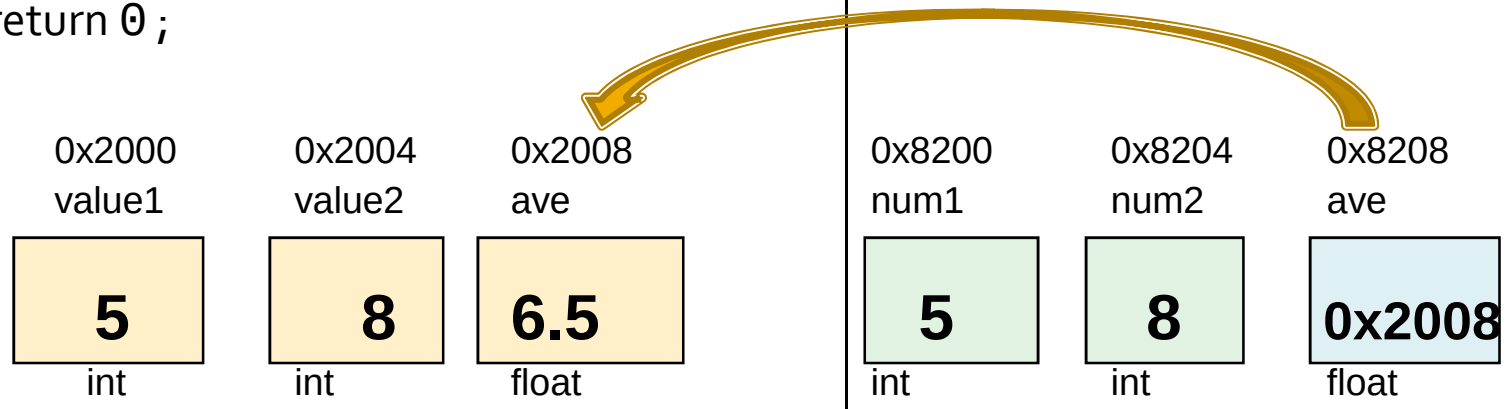
```
AverageTwo (int  num1, int  num2, float *ave)
{


  *ave = (num1 + num2) / 2.0 ;
  return;
}
```



Parameter Passing by Reference and Local Variables

```
#include <stdio.h>
AverageTwo (int num1, int  num2, float *ave) ;

int main ( )
{
     float ave ;
     int value1 = 5, value2 = 8 ;

     AverageTwo (value1, value2, &ave)  ;
     ...
    return 0 ;
}
```

```
AverageTwo (int  num1, int  num2, float *ave)
{

  *ave = (num1 + num2) / 2.0 ;
  return;
}
```



Parameter Passing by Reference and Local Variables

```
#include <stdio.h>
AverageTwo (int num1, int  num2, float *ave) ;

int main ( )
{
     float ave ;
     int value1 = 5, value2 = 8 ;

     AverageTwo (value1, value2, &ave)  ;
     ...
    return 0 ;
}
```

```
AverageTwo (int  num1, int  num2, float *ave)
{


  *ave = (num1 + num2) / 2.0 ;
  return;
}
```