



Programming Studio #8

ECE 190



Programming Studio #8

- Announcements
- Concepts this week
 - Loops
 - Functions
 - GDB
- Exercise: debugging



Announcements

- Exam 1 regrade requests due today
- MP3.1 due Wednesday (3/17) by 5p (Start now)




Loops

- Loops in C used to replace branched loops in assembly
- 3 loop constructs in C
 - while
 - do ... while
 - for

while Loop

- Syntax:

```
while (condition) {  
    <execute commands enclosed by braces (i.e.,  
    loop code>  
    <condition update>  
}
```

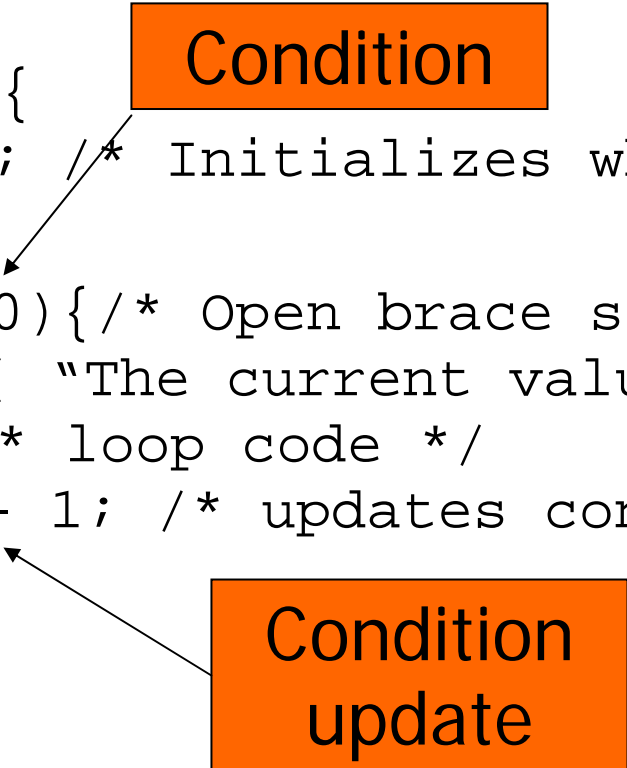


Important!

- Wrapped commands execute only if condition true
- As long as condition true, commands within braces executed repeatedly

while Example

```
int main () {  
    int a = 5; /* Initializes while loop */  
    while (a>0){ /* Open brace starts loop code */  
        printf( "The current value of a is %d\n", a );  
        /* loop code */  
        a = a - 1; /* updates condition */  
    }  
}
```



- How many lines will this code print?

do ... while Loop

- do ... while loops similar to while
- One exception: do ... while loops ***always*** execute code in braces at least ***once***

for Loop

- for loop most straight-forward loop
- Syntax: (cvar short for condition variable)

```
for(<init cvar>; <condition>; <update cvar>) {  
    <loop code>}
```
- Order of execution
 1. Initialize
 2. Check condition
 3. Execute loop code
 4. Update condition
- Repeat 2-4 until 2 fails

for Example

```
int main () {  
    int a;  
    for(a=5; a>0; a--)//*Initializes the while loop*/  
    { /* Open brace loop code */  
        printf( "The current value of a is %d\n", a );  
        /* loop code */  
    }  
}
```

- This code results in the same output as the while example code

Functions

- Same as subroutines

- Declared like variables: e.g.,

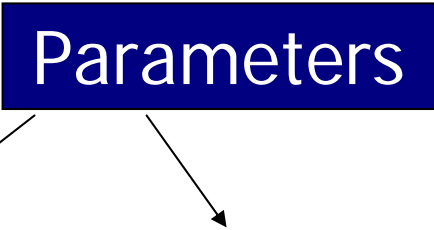
```
int func1(); char func2();
```



Return types

- Declaration specifies return value type (i.e., output)
func1() – returns integer
func2() – returns character

Function Parameters

- Functions often take parameters 
 - Ex.: `int larger_num(int num1, int num2);`
 - num1 and num2 are integers and are passed to larger_num
 - Ex.: `int a = larger_num(5,6);`
- Function larger_num then manipulates num1 and num2 to output larger of two

Function: larger_num

```
int larger_num( int num1, int num2 );  
void main () {  
  int a = larger_num(5,6);  
}
```

Call

Declaration
(must declare before call)

Implementation

larger_num() written outside main()

```
int larger_num(int num1, int num2) {  
  if( (num1-num2) > 0) return num1;  
  else if( (num2 - num1) > 0) return number2;  
  else return number1; /* both numbers are equal,  
    therefore return either */  
}
```



Debugging – gdb

- Use gdb to debug C
- Compile using debug flag (i.e., -g)
`gcc -g <C source file> -o <executable name>`
- Run gdb in terminal – `gdb <executable name>`
- Insert breakpoints – `break <line number>`
- Run program – `run`
- Step through program and into functions – `step`
- Step through program – `next`
- Continue program – `continue`

Programming Exercise

- Download buggy code

wget

`http://courses.ece.illinois.edu/ECE190/discussion/spring10/ps08/odd_or_even.c`

- `main()` takes int value from user
- Function manipulates int value as follows
 - Output factorial if odd
 - Output square of input if even
- Correct compile bugs using gcc warnings
- Correct run-time bugs using gdb