ISA 563: Fundamentals of Systems Programming

Control Flow: Decision & Repetition Statements

Jan 22, 2013

Outline

- Review
  - Keywords
  - Basic data types, arrays, string handling
  - Command line arguments
- Control Flow (break, return, continue)
- Decision Statements (if, else, switch, case)
  - Boolean expressions
- Repetition Statements (for, while)
  - Array processing

Keywords in C

`if, else, switch, for, do`

`break, continue, goto, case, default`

`int, float, double, char`

`long, short, signed, unsigned, register, const, volatile, extern, static, auto`

`typedef, struct, union, enum, sizeof`

`return, void`

Control Flow

- The actual sequence of instructions executed
- Not necessarily the order of the source listing
- Groups of related code go into statement blocks
- { }

Changing Control Flow

- Predicated on the evaluation of a boolean expression or explicit keyword
- Three ways to change control flow:
  - Decide on a choice between alternatives
  - Repeat the current block of statements
  - Unconditional jump

Boolean Expressions (review)

- Boolean expressions are any valid C expression that evaluates to an integer value
- The value zero is taken to mean 'false'
    - Any other value is 'true', although 1 (one) is used most often by convention
- Programs can make a decision between two different flows of control based on the result of a boolean expression
    - Also based on the value of computation

if

- The 'if' keyword is an operator that evaluates a boolean expression and conditionally executes the code of the statement block immediately following the 'if' if the condition evaluates to 'true':

```
if(expression)
{
  // code to execute if expression is true
}
```

else

- If 'if' statement evaluates to 'false', then the code statements in the body of the 'if' are note executed.
    - Instead, control flow 'falls through' the if
- Sometimes, we want to execute code if the condition is false. This is accomplished with 'else':

```
if (condition) {
    // code1
} else {
    // code2
}
```

switch

- The switch statement allows you to pick from different cases:

```
int data_value = 0;
switch ( data_value ) {
  case 0:
    // do something
    break;
  case 1:
    // do something else
    break;
  default:
    // do safe thing
}
```

Looping and Repetition

- Often, you want to execute the same set of statements multiple times
  - Reading input
  - Drawing graphics
  - Calculating something
- Need a way to 'loop' or repeat
  - Loop control variable
  - Initialization
  - Increment/decrement/loop control maintenance
  - condition

while

- The while statement allows for looping while a condition is true

```
while(1)
{
   // loop forever
}
```

```
int counter = 0;
int limit = 10;

while(counter < limit)
{
   printf("counter == %d\n", counter);
   counter++;
}
```

for

- The 'for' statement is like 'while' but gathers the bookkeeping work into a single statement

```
limit = 10;
for(counter=0; counter < limit; counter++)
{
  printf(“counter == %d\n”, counter);
}
```