C STATEMENTS CONT'D

- recall that so far we have covered
    - Selection Statements:
        - if, if/else, if/else if/else
    - Iteration Statements:
        - for, while, do/while
    - Jump Statements:
        - return

- switch
    - another selection statement with multiple branches is the switch statement.
    - it has the general form:

```
switch (expression)
{
    case constant1 :
         :
       break;
    case constant2 :
         :
       break;
     :
    default :
         :
}
```

must evaluate to an int or char type

expression is compared against these constant values

when a match occurs, the corresponding statements of the matching case are executed, followed by a break statement, which exits the switch statement

if no match is found, the default case is selected and since it comes last, no break is needed.

EXAMPLE:

```
printf("press a key:");
ch= getchar();
switch (ch)
{
    case 'a':
      printf("a is for apple");
      break;
   case 'b':
      printf(" b is for bee");
      break;
   case 'c':
      printf("c is for me\n");
      printf("get it?");
      break;
  default:
      printf("that is all, folks!");
}
```

Nesting

- note that statements we have covered so far can be <u>nested</u>, that is, statements of one kind can be placed within statements of the same kind.
- for example, a nested for loop would be:

```
int sum;
for (int i=0; i!=10; ++i)
{
    sum=0;
    for (int j=0; j!=20; ++j)
    {
        sum += j*i + 3;
    }
    :
}
```

- a nested if statement could look like:

```
if (a==3 && b==7)
{
    if (c== 22 || d > 12.3)
    {
        if (e <((7 * f + 3) % 2 )
        {
                 :
        }
    }
    else
    {
            :
    }
}
else if (a== 4 && b== 8)
{
        :
}
else
{
      :
}
```

Infinite Loops

- can be useful in certain situations, particularly in embedded systems design
- an infinite loop can be realized using any iteration statement:
    - an infinite for loop:

      ```
      for ( ; ;)
      ```

      example:

      ```
      for ( ; ;)
          printf ("This statement runs forever! \n");
      ```
    - an infinite while loop:

      ```
      while (1)
      {
          :
      }
      ```

      example:

      ```
      while (1)
          printf ("This statement runs forever! \n");
      ```

- we can use a break statement to exit an (otherwise) infinite loop:

```
ch='\0';
for (;;)
{
    ch=getchar(); /* get a character from the keyboard */
    if (ch=='y' || ch=='n')
        break;  /* exit the loop */
}

switch (ch)
{
    case 'y':
      printf ("You typed YES!");
      break;
   case 'n':
      printf("you answered NO!");
      break;
}
```

for loops with no bodies

- often used in embedded programming to create delays, for example

```
#define     DELAY      1048576
                :
printf ("delay loop starting!\n");
for (int t=0; t!= DELAY; ++t);
printf ("delay loop ended!\n");
```

note that declaring a variable within a loop is not permitted in C89, but allowed in C99 (and C++).

- in-class lab / homework:
    - compose a C program that demonstrates the examples so far as a series, but don't get stuck in an infinite loop!

exit()

- found in stdlib.h, you can use the function exit() to break out of your program
- this is often used to indicate an error, and you can use the argument of exit() to indicate the error code.

  for example:

```
int exit_code = 0;
if ( !graphics_card() )
{
    exit_code = 1;
    exit (exit_code);
}
else if (!joy-stick_found() )
{
    exit_code = 2;
    exit (exit_code);
}
```

- the stdlib library also contains the <u>macros</u> EXIT_SUCCESS and EXIT_FAILURE which can be used as arguments (return codes) for exit(), as well

FUNCTIONS

- the general form of a C function is:

```
return-type function-name (parameter list)
{
        :
     return expression;
}
```

expression must be of type return-type; if return-type is void, then there is no expression

- examples of functions include:

```
int  decrement (int arg)
{
    arg--;
    return arg;
}
```

arg is a function local variable; if a global variable happens to have the same name, the local variable is assumed inside the function

```
int  is_found (char *s, char c)
{
   /* return 1 if c found in array s, Ø otherwise */
   while (*s)
      if (*s == c)
         return 1;  /* function returns early */
      else
         s++;
   return Ø;
}
```

an array without the brackets, is a pointer;
e.g.
int s[5];
:
int *p;
p=s;
now, p points to the array.

```
void  delay (int amount)
{
   for (int t = 0; t != amount; ++t);
   return;
}
```

- a function must be declared, before main() as follows:

```
#include <stdio.h>
#include <stdlib.h>

/* function prototypes */
int decrement (int );
void delay (int );

int main ()
{
        :
    return 0
}

/* function bodies */
int decrement (int arg)
{
    return --arg;
}

int delay (int amount)
{
    int counter = amount;
    while (counter != 0)
    {
        -- counter;
    }
}
```

usually just the parameter types are needed

- HOMEWORK:
  - write a function to return the average of the values in an integer array
  - write a function to find median value of an array of sorted numbers (numbers are stored in ascending order)

    Note:
    - for an even number of elements, the median is the average of the two elements closest to the array middle.
    - for an odd number of elements, the median is simply the middle element.