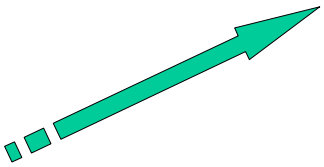
Lecture 8



1. Introduction
2. Binary Representation
3. Hardware and Software
4. High Level Languages
5. Standard input and output
6. Operators, expression and statements
7. Making Decisions
8. Looping
9. Arrays
10. Basics of pointers
11. Strings
12. Basics of functions
13. More about functions
14. Files
14. Data Structures
16. Case study: lottery number generator

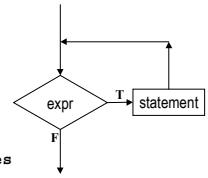
Looping

- Last lecture we saw how we could follow different paths through a program using decisions.
- This time we will look at repeating sections of code several times
- C has 3 types of loop
  - `while` loops
  - `do...while` loops
  - `for` loops

while loop

```
while(expr)
    statement;
```

```
while(expr){
    statement;
    // More lines
    // of code
}
```



- As with the if statement, expr can be true (non-zero) or false (zero).
- If true, statement is executed. This can be a block of statements enclosed in braces { }
- NOTE: the statement could be executed zero times

while loop

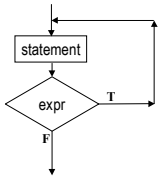
- e.g., `while(hungry)`
  `eat_sandwich();`
- If your not hungry you eat zero sandwiches

while.c

```
/* Example: while loop */

#include <stdio.h>

main()
{
  int m = 0, n;

  puts("Enter a small integer (e.g. 10): ");
  scanf("%i", &n);

  while (m < n)
  {
    putchar('*');
    m++;
    if (m % 20 == 0)  /* newline after every 20 */
      putchar('\n');
  }

  putchar('\n');
}
```

do…while loop

```
do {
     statement;
     // More statements if you like
} while (expr);
```



- Again multiple statements can be included in the { }
- Note that statement is executed at least once
- e.g.,

```
do {
   take_driving_test();
} while (!passed);
```

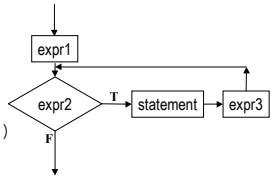
In this case you have to take the test once, maybe more times, until you pass

do…while loop

dowhile.c

```
/* Example: do-while loop */

#include <stdio.h>

main()
{
  int m;

  do {
       puts("Enter an integer (0 to finish):");
       scanf("%i", &m);
       printf("You entered %i\n\n", m);
     }
       while (m);

  puts("Finished");
}
```

for loop




```
for(expr1;expr2;expr3)
      statement;
```

- Again the statement can be a block
- The `for` loop is the most flexible type of loop.
- It is most often used when you want to do something n times e.g.,

```
for(int i=0;i<n;i++)
      do_something;
```

initialisation expression — conditional expression — incremental expression

- You can create any type of loop from a for loop plus achieve some other useful functions if your creative

for.c

```
/* Example: for loops (also a simple macro) */

#include <stdio.h>

#define skip {putchar('\n'); putchar('\n');}
/* This is called a "macro" */

main()
{
  int m, i, n = 20;

  /* Standard C idiom for doing something n times: */

  for (i = 0; i < n; i++)
    putchar('*');

  skip;

  for (m = 1; m <= 10; m++)  /* counting up */
    printf("%i ", m);

  skip;

  for (m = 10; m > 0; m--)  /* counting down */
    printf("%i ", m);

  skip;

  for (m = 0; m < 21; m += 2) /* counting up in 2s */
    printf("%i ", m);

  putchar('\n');
}
```

forcomma.c

```
/* Example: comma operator in for loops */

#include <stdio.h>

main()
{
  int i, j;

  for (i = 0, j = 9; i < 10; i++, j--)
    printf("%i plus %i is %i\n", i, j, i + j);

  putchar('\n');
}
```

Nested for loops

- We've seen nested `if` statements, its also very common to see nested `for` loops

```
/*  Example: nested for loops */

#include <stdio.h>

main()
{
  int row; /* Row index */
  int col; /* Column index */

  puts("200 X's:\n");

  for (row = 0; row < 10; row++)  /* outer loop (rows) */
  {
    for (col = 0; col < 20; col++)  /* inner loop (columns) */
      putchar('X');
    putchar('\n');
  }


  puts("\nMultiplication table:\n");
```

```
  for (row = 0; row < 11; row++)  /* outer loop (rows) */
  {
    if (row == 0) /* heading */
    {
      printf(" X |");
      for (col = 1; col < 11; col++)  /* inner loop (columns) */
        printf("%4i", col);
      putchar('\n');

      printf("----+");
      for (col = 1; col < 42; col++)  /* inner loop (columns) */
        putchar('-');
      putchar('\n');
    }
    else /* normal rows */
    {
      for (col = 0; col < 11; col++)  /* inner loop (columns) */
      {
        if (col == 0)
          printf("%2i |", row);
        else
          printf("%4i", row * col);
      }
      putchar('\n');
    }
  }
}
```

```
200 X's:

XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX

Multiplication table:

 X |   1   2   3   4   5   6   7   8   9  10
---+----------------------------------------
 1 |   1   2   3   4   5   6   7   8   9  10
 2 |   2   4   6   8  10  12  14  16  18  20
 3 |   3   6   9  12  15  18  21  24  27  30
 4 |   4   8  12  16  20  24  28  32  36  40
 5 |   5  10  15  20  25  30  35  40  45  50
 6 |   6  12  18  24  30  36  42  48  54  60
 7 |   7  14  21  28  35  42  49  56  63  70
 8 |   8  16  24  32  40  48  56  64  72  80
 9 |   9  18  27  36  45  54  63  72  81  90
10 |  10  20  30  40  50  60  70  80  90 100
```

continue, break and goto

Bad programming practice

- Sometimes we want to jump to the end of a loop
  - for all 3 types we can do this with `continue;`
- Often we want to leave a loop early
  - for all 3 types we can do this with `break;`
- One common programming technique is to construct an infinite loop (one that does not terminate) but jump out of it using a break statement
- C also includes a `goto` statement but I am not going to explain how to use it, its bad practice and shouldn't have been included in the programming language (in my opinion)
  - I have never used goto in either a C or C++ program and that's many hundreds of thousands of lines of code

continue.c

```
/* Example: skipping to end of loops using continue */

#include <stdio.h>

main()
{
  int h = 500, count = 0;

  printf("All the numbers from %i to zero\n", h);
  puts("not divisible by 2, 3, 5 or 7:\n");

  do {
      if (h % 2 == 0)  /* skip those divisible by 2 */
        continue;
      if (h % 3 == 0)  /* ... by 3 */
        continue;
      if (h % 5 == 0)  /* ... by 5 */
        continue;
      if (h % 7 == 0)  /* ... by 7 */
        continue;
      printf("%5i", h), count++;
      if (count % 10 == 0)  /* print 10 per line */
        putchar('\n');
    }
    while (h--);
}
```

break.c

```
/* Example: exiting loops using break */

#include <stdio.h>

main()
{
  int s;

  puts("Infinite for loop:");
  for ( ; ; )
  {
    printf("Enter an integer (999 to finish): ");
    scanf("%i", &s);
    if (s == 999)
      break;
  }
  puts("Loop exited\n");

  puts("Infinite while loop:");
  while (1)
  {
    printf("Enter an integer (999 to finish): ");
    scanf("%i", &s);
    if (s == 999)
      break;
  }
  puts("Loop exited\n");

  puts("Infinite do-while loop:");
  do {
      printf("Enter an integer (999 to finish): ");
      scanf("%i", &s);
      if (s == 999)
        break;
    }
    while (1);
  puts("Loop exited\n");
}
```

Menu Example

- You now know enough C to write some real life programs. As an example here is a program which provides a menu system

```
McDUCK'S ELECTRONIC MENU SYSTEM
===============================

Smile at customer; "How may I help you?"

   [B]ig Duck      - press B (then Enter)
   [H]amburger     - press H (then Enter)
   [C]heeseburger  - press C (then Enter)
   [F]ries         - press F (then Enter)
   [L]arge fries   - press L (then Enter)
   [S]oft drink    - press S (then Enter)
   [A]pple pie     - press A (then Enter)
   [M]ilk shake    - press M (then Enter)

Press [X] (then Enter) when order is complete
```

```
b
  Big Duck      $2.50
r
  Large fries   $0.75
s
  Soft drink    $0.50
x

Order complete - 3 items - total $3.75

"Thank you! That'll be $3.75"
Take money, give change
"Thank you!"


Fetch 3 items
"Thanks for eating at McDuck's! Enjoy your meal!"
```

```
/* Example: menu system, using do...while, getchar and switch */

/* You've reached a point in the course where you know enough C
to start writing real programs. Here's a fairly realistic example:
a menu system. I could have used functions for this program, there's
no way to cancel an item. Also, when we get on to functions, the
structure could be made more modular. Nevertheless, it works! */

#include <stdio.h>

#define BIG 2.50        /* price of Big Duck */
#define HAM 2.00        /* price of hamburger */
#define CHE 2.00        /* price of cheeseburger */
#define FRI 0.50        /* price of fries */
#define LAR 0.75        /* price of large fries */
#define SOF 0.50        /* price of soft drink */
#define APP 1.00        /* price of apple pie */
#define MIL 0.75        /* price of milk shake */

int main(void)
{
  char key;             /* key pressed */
  int items = 0;        /* number of food items */
  int done = 0;         /* order complete? */
  float total = 0.00;   /* total price */

  /* Output information: */

  puts("McDUCK'S ELECTRONIC MENU SYSTEM");
  puts("================================\n");

  puts("Smile at customer; \"How may I help you?\"\n");

  puts("   [B]ig Duck      - press B (then Enter)");
  puts("   [H]amburger     - press H (then Enter)");
  puts("   [C]heeseburger  - press C (then Enter)");
  puts("   [F]ries         - press F (then Enter)");
  puts("   [L]arge fries   - press L (then Enter)");
  puts("   [S]oft drink    - press S (then Enter)");
  puts("   [A]pple pie     - press A (then Enter)");
  puts("   [M]ilk shake    - press M (then Enter)\n");
  puts("Press [X] (then Enter) when order is complete\n");
```

```
/* Handle key presses: */
  do {
    fflush(stdin);  /* flush keyboard buffer */
    key = getchar(); /* read key */
    switch (key)
    {
      case 'B': case 'b':
        printf("  Big Duck    $%4.2f\n", BIG);
        items++;
        total += BIG;
        break;

      case 'H': case 'h':
        printf("  Hamburger   $%4.2f\n", HAM);
        items++;
        total += HAM;
        break;

      case 'C': case 'c':
        printf("  Cheeseburger $%4.2f\n", CHE);
        items++;
        total += CHE;
        break;

      case 'F': case 'f':
        printf("  Fries       $%4.2f\n", FRI);
        items++;
        total += FRI;
        break;

      case 'L': case 'l':
        printf("  Large fries  $%4.2f\n", LAR);
        items++;
        total += LAR;
        break;

      case 'S': case 's':
        printf("  Soft drink   $%4.2f\n", SOF);
        items++;
        total += SOF;
        break;

      case 'A': case 'a':
        printf("  Apple pie    $%4.2f\n", APP);
        items++;
        total += APP;
        break;
```

```
      case 'M': case 'm':
        printf("  Milk shake   $%4.2f\n", MIL);
        items++;
        total += MIL;
        break;

      case 'X': case 'X':
        printf("\nOrder complete - ");
        printf("%i items - total $%4.2f\n", items, total);
        done = 1;
        break;

      default:
        puts("  Invalid key! Try again.");
        break;
    }
  } while (!done);

  /* Output information: */

  printf("\n\"Thank you! That'll be $%4.2f\"\n", total);
  puts("Take money, give change");
  puts("\"Thank you!\"\n");
  printf("\nFetch %i items\n", items);
  puts("\"Thanks for eating at McDuck's! Enjoy your meal!\"");
}
```