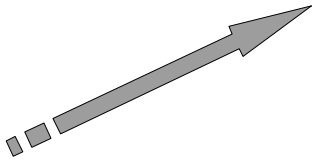
Lecture 5



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

Standard Input and Output

- Many programs have the form:

Input Data
Process It
Output Data ⇩

- Where data can be read from the keyboard or a file and be printed to the screen or a file

Standard Input and Output

- In C, the *standard input* (stdin) is generally connected to the keyboard and the *standard output* (stdout) to the screen.
- However, in UNIX we can redirect the either or both at run time from the command line
- If we have a simple program a.out which takes some text from the keyboard and prints the results to the screen, then

```
a.out > outfile.txt                - sends output to file
a.out < infile.txt                 - reads input from file
a.out < infile.txt > outfile.txt   - does both
prog1 | prog2                      - sends output of prog1 to input of prog2
```

Standard Input and Output

- Luckily some nice person has written a whole library of C calls which allow use to easily perform stdio i.e. <stdio.h>
- In order to use these routines we need to include the appropriate header file

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

- We will study some of these more common routines, namely
  - OUTPUT: `puts, printsf and putchar`
  - INPUT: `scanf, getchar`

hello.c

```
/* Hello world program */

#include <stdio.h>

void main()
{
   printf("Hello world");
}
```

stdout

- The first program we will look at uses `puts` to *put a string* to standard out
  - a newline is automatically added to the end of the string
  - see `puts.c` (available on web)

```
/* Example: outputting strings using puts */
#include <stdio.h>
void main()
{
  puts("To be or not to be: that is the question:");
  puts("Whether 'tis nobler in the mind to suffer");
  puts("The slings and arrows of outrageous fortune,");
  puts("Or to take arms against a sea of troubles,");
  puts("And by opposing end them? To die: to sleep;");
  puts("No more; and by a sleep to say we end");
  puts("The heart-ache and the thousand natural shocks");
  puts("That flesh is heir to, 'tis a consummation");
  puts("Devoutly to be wished. To die, to sleep;");
  puts("To sleep: perchance to dream: ay, there's the rub.");
  puts("For in that sleep of death what dreams may come");
  puts("When we have shuffled off this mortal coil,");
  puts("Must give us pause.");
  /* Hamlet */
}
```

stdout

- Often we need to print data that the program has calculated and we need to control the formatting of this data.
  - e.g. the value 30 could be printed as 30 or 30.00 or +3E+01
  - this can all be done with `printf` which prints formatted output to standard out
  - note: `printf` does not automatically add a new line for us
- Basic printing
  - for an int j: printf("%i",j);
  - for a float x: printf("%f",x);
- The %-string is a format conversion string

printf1.c

```
/* Example: outputting numeric data using printf */
#include <stdio.h>
void main()
{
  int j = 5;
  float x = 123.4567890123456789;
  double z = 123.4567890123456789;
  char c = 'A';

  printf("The value of j is %i\n\n", j);

  printf("The value of x is %f\n", x);
  printf("...or %E\n", x);
  printf("...or %30.27f\n\n", x);

  printf("The value of z is %f\n", z);
  printf("...or %30.27f\n\n", z);

  printf("\nThe value of c is %c or %i\n", c, c);
  c++;
  printf("The new value of c is %c or %i\n\n", c, c);

  printf("0.05 is equivalent to %i%%\n\n", j);

  printf("Hello ");
  printf("world!\n");
}
```

```
The value of j is 5

The value of x is 123.456787
...or 1.23456E+002
...or 123.456787109375000000000000000

The value of z is 123.456789
...or 123.456789012345680000000000000


The value of c is A or 65
The new value of c is B or 66

0.05 is equivalent to 5%

Hello world!
```

printf2.c

```
/* Example: formatting integer data using printf */
#include <stdio.h>

main()
{
  int j = 1234;
  int k = -5678;

  puts("j = 1234");
  puts("k = -5678");

  puts("\n|...| is used to show the field width.\n");

  printf("Using %i,    j = |%i|\n", j);
  printf("             k = |%i|\n\n", k);

  printf("Using %i+i,  j = |%+i|\n", j);
  printf("             k = |%+i|\n\n", k);

  printf("Using %% i,  j = |% i|\n", j);
  printf("             k = |% i|\n\n", k);

  printf("Using %8i,   j = |%8i|\n", j);
  printf("             k = |%8i|\n\n", k);

  printf("Using %X-8i, j = |%-8i|\n", j);
  printf("             k = |%-8i|\n\n", k);

  printf("Using %X- 8i, j = |%- 8i|\n", j);
  printf("             k = |%- 8i|\n\n", k);

  printf("Using %X08i, j = |%08i|\n", j);
  printf("             k = |%08i|\n\n", k);

  printf("Using %X8.6i, j = |%8.6i|\n", j);
  printf("             k = |%8.6i|\n\n", k);
}
```

```
j = 1234
k = -5678

|...| is used to show the field width.

Using %i,    j = |1234|
             k = |-5678|

Using %+i,   j = |+1234|
             k = |-5678|

Using % i,   j = | 1234|
             k = |-5678|

Using %8i,   j = |   1234|
             k = |  -5678|

Using %-8i,  j = |1234    |
             k = |-5678    |

Using %- 8i, j = | 1234    |
             k = |-5678    |

Using %08i,  j = |00001234|
             k = |-0005678|

Using %8.6i, j = |  001234|
             k = | -005678|
```

printf3.c

```
/* Example: formatting float/double data using printf */

#include <stdio.h>

main()
{
  double a = 123.4;  /* These could all be floats */
  double b = -567.8;
  double c = 987654321;
  double d = -0.00008765;

  puts("a = 123.4");
  puts("b = -567.8");
  puts("c = 987654321");
  puts("d = -0.00008765");

  puts("\n|...| is used to show the field width.\n");

  printf("Using %f,     a = |%f|\n", a);
  printf("              b = |%f|\n", b);
  printf("              c = |%f|\n", c);
  printf("              d = |%f|\n\n", d);

  printf("Using %8.3f,  a = |%8.3f|\n", a);
  printf("              b = |%8.3f|\n", b);
  printf("              c = |%8.3f|\n", c);
  printf("              d = |%8.3f|\n\n", d);

  printf("Using %E,     a = |%E|\n", a);
  printf("              b = |%E|\n", b);
  printf("              c = |%E|\n", c);
  printf("              d = |%E|\n\n", d);

  printf("Using %12.3E, a = |%12.3E|\n", a);
  printf("              b = |%12.3E|\n", b);
  printf("              c = |%12.3E|\n", c);
  printf("              d = |%12.3E|\n\n", d);

  printf("Using %G,     a = |%G|\n", a);
  printf("              b = |%G|\n", b);
  printf("              c = |%G|\n", c);
  printf("              d = |%G|\n\n", d);
}
```

```
a = 123.4
b = -567.8
c = 987654321
d = -0.00008765

|...| is used to show the field width.

Using %f,     a = |123.400000|
              b = |-567.800000|
              c = |987654321.000000|
              d = |-0.000088|

Using %8.3f,  a = | 123.400|
              b = |-567.800|
              c = |987654321.000|
              d = |  -0.000|

Using %E,     a = |1.234000E+002|
              b = |-5.678000E+002|
              c = |9.876543E+008|
              d = |-8.765000E-005|

Using %12.3E, a = |  1.234E+002|
              b = | -5.678E+002|
              c = |  9.877E+008|
              d = | -8.765E-005|

Using %G,     a = |123.4|
              b = |-567.8|
              c = |9.87654E+008|
              d = |-8.765E-005|
```

printf.bug

```
/* BUG ZONE!!!
Example: outputting numeric data using printf */

#include <stdio.h>

main()
{
  int j = 5;
  float x = 123.4567890123456789;
  double z = 123.4567890123456789;
  char c = 'A';

  printf("The value of j is %f\n\n", j); /* BUG */

  printf("The value of x is %i\n", x); /* BUG */

  printf("The value of z is %c\n", z); /* BUG */

  printf("\nThe value of c is %f or %s\n", c, c); /* 2 BUGS */

  printf("0.05 is equivalent to 5%"); /* BUG */
}
```

```
The value of j is 0.000000

The value of x is 0
The value of z is Ó

The value of c is 0.000000 or
0.05 is equivalent to
```

puchar

- `putchar` : put a character (to stdout)
  - Remember a char can be treated as a character or as a number so `putchar('A');` and `putchar(65);` are equivalent

```
/* Example: outputting characters using putchar */

#include <stdio.h>

main()
{
  putchar('H');
  putchar('e');
  putchar(108);
  putchar(108);
  putchar('o');
  putchar('!');
  putchar('\t');
  putchar('*');
  putchar('\n');
}
```

```
Hello!   *
```

Can you spot the bugs in this program ?

```
/*  BUG ZONE!!!
Example: standard output */

#include (studio.h)                /* 2 BUGS! */

Main()                            /* BUG! */
{
  puts(Multiplication);           /* BUG! */

  printf("9 times 7 = %c", 9 * 7);  /* BUG! */

  putchar("\n");                  /* BUG */

  print("9 times 8 = %i", 9 * 8);   /* BUG */

  printf("/n");                   /* BUG */
}
```

stdin

- `scanf` : scan formatted (from stdin)
- Basic use
  - for an int j : `scanf("%i",&j);`
  - for a float x : `scanf("%f",&x);`
- the & is very important and must not be omitted (an easy mistake to make)!
  - &x means "the address of x", well see why later
- Again the %-string is a format conversion string.

scanf.c

```
/* Example: inputting numeric data using scanf */

#include <stdio.h>
#include <limits.h> /* defines INT_MIN, INT_MAX, LONG_MIN, LONG_MAX */

main()
{
  int j;
  long int k;
  float x;
  double z;

  printf("Enter an integer (between %i and %i): ", INT_MIN, INT_MAX);
  scanf("%i", &j);
  printf("You entered %i\n\n", j);

  printf("Enter a long integer (between %li and %li): ", LONG_MIN, LONG_MAX);
  scanf("%li", &k);
  printf("You entered %li\n\n", k);

  printf("Enter a floating point number: ");
  scanf("%f", &x);
  printf("You entered %20.10E\n\n", x);

  printf("Enter a double precision floating point number: ");
  scanf("%lf", &z);
  printf("You entered %20.10E\n\n", z);

  puts("\n\nTry again: enter invalid data and see what happens!");
}
```

```
Enter an integer (between -2147483648 and 2147483647): 10
You entered 10

Enter a long integer (between -2147483648 and 2147483647): 10
You entered 10

Enter a floating point number: 1.1
You entered    1.1000000238E+000

Enter a double precision floating point number: 1.1
You entered    1.1000000000E+000


Try again: enter invalid data and see what happens!
```

scanf.bug

```
/* BUG ZONE!!!
Example: inputting numeric data using scanf */

#include <stdio.h>
#include <limits.h> /* defines INT_MIN, INT_MAX */

main()
{
  int j;
  double z;

  printf("Enter an integer (between %i and %i): ", INT_MIN, INT_MAX);
  scanf("%i", j); /* BUG */
  printf("You entered %i\n\n", j);

  printf("Enter a double precision floating point number: ");
  scanf("%i", &z); /* BUG */
  printf("You entered %20.10E\n\n", z);
}
```

getchar

- `getchar` : get a character (from stdin)
- The input is buffered - this means you have to press ENTER after the character.
- It can also cause problems because this ENTER (='\n') will be read next time getchar is called.
- The solution is to "flush the buffer" before reading it using `fflush(stdin);`

getchar.c

```
/*  Example: getting a single character from the keyboard, using getchar */

#include <stdio.h>

main()
{
   char key;

  printf("Press a key (then ENTER): ");
  key = getchar();
  printf("You pressed %c\n", key);
  puts("-------------");

  printf("Press another key: ");
  key = getchar();
  printf("You pressed %c\n", key);
  puts("-------------");

  puts("Oops! What went wrong?");
  puts("Let's try again...\n\n");

  printf("Press a key (then ENTER): ");
  fflush(stdin); /* flush the keyboard buffer */
  key = getchar();
  printf("You pressed %c\n", key);
  puts("-------------");

  printf("Press another key: ");
  fflush(stdin); /* flush the keyboard buffer */
  key = getchar();
  printf("You pressed %c\n", key);
  puts("-------------");

  puts("Ah, that's more like it!");
  /* getchar is both echoed and buffered */
}
```

```
Press a key (then ENTER): p
You pressed p
-------------
Press another key: You pressed

-------------
Oops! What went wrong?
Let's try again...


Press a key (then ENTER): p
You pressed p
-------------
Press another key: r
You pressed r
-------------
Ah, that's more like it!
```