

Printing (to terminal): `printf`

- general syntax:

```
printf("a char: %c, a new line:\n", cval);
printf("an int: %d, a tab:\t", ival);
printf("a long: %ld, a vertical tab:\v", lval);
printf("a float: %f, another float: %g, another float: %G, a backspace:\b",
      fval1, fval2, fval3);
printf("another float: %e, another float: %E, an alert:\a",
      fval4, fval5);
printf("a double: %f, another double: %g, another double: %G:\n" \
      "another double: %e, another double: %E:\n",
      dval1, dval2, dval3, dval4, dval5);
printf("a pointer: %p\n", &ival);
```

- note the usage of line continuation "`\`" mark above
- there are plenty of tricks you could try, e.g.

```
printf("an int: %.2d\n", ival);
printf("an int: %*d\n", width, ival);
```

- for details see the `printf` manual on the web

Scanning (from terminal): scanf

- general syntax:

```
printf("enter a char: ");
scanf("%c", &cval);
printf("you entered: %c\n", cval);
printf("enter an int: ");
scanf("%d", &ival);
printf("you entered: %d\n", ival);
printf("enter a float: ");
scanf("%f", &fval);
printf("you entered: %f\n", fval);
printf("enter a double: ");
scanf("%lf", &dval);
printf("you entered: %f\n", dval);
```

- note how you scan a double, its weird, isn't it?
- one could use the function `getchar` instead of using `scanf` for reading character (`scanf` preferred)

```
printf("enter another char: ");
cval = getchar( );
printf("you entered: %c\n", cval);
```

Control Flow I

- for loop example:

```
int ii, nn = 5;
double val = 0.0;
```

```
for (ii = 1; ii <= nn; ii = ii + 1) {
    val = val + ii;
}
```

- some short hands:

```
int ii, nn = 5;
double val = 0.0;
```

```
for (ii = 1; ii <= nn; ++ii) {
    val += ii;
}
```

Aside

- `++ii` is same as `ii = ii + 1`
 - DO NOT use `ii++`, unless you know what you are doing, would clarify the difference between `++ii` and `ii++` later
- `ii += 1` is same as `ii = ii + 1`
- in general, `ii += whatever` is same as `ii = ii + whatever`
- more generally, we have constructs like `ii -= whatever`,
`ii *= whatever` and `ii /= whatever`
- completely unrelated but:
 - “%” is the modulus operator
 - there is no “^” or the power operator, one needs to use the `pow` function by including the math library: `#include <math.h>`
 - * note in this case compile with `gcc -lm prog-name.c`

Control Flow III

- while loop example:

```
int ii = 1, nn = 5;
double val = 0.0;
```

```
while (ii <= nn) {
    val += ii;
    ++ii;
}
```

- do-while loop example:

```
int ii = 1, nn = 5;
double val = 0.0;
```

```
do {
    val += ii;
    ++ii;
} while (ii <= nn);
```

– note the “;” at the end of while

Control Flow IV

- if-else statement example:

```
if ((1 <= ii) && (ii < 3)) {  
    printf("low\n");  
}  
else if ((3 <= ii) && (ii < 5)) {  
    printf("medium\n");  
}  
else {  
    printf("high\n");  
}
```

- && stands for “logical and”
- || stands for “logical or”
- == stands for “equal”
- != stands for “not equal”

Control Flow V

- switch statement example:

```
switch (ii) {
  case 1:
  case 2:
    printf("low\n");
    break;

  case 3:
  case 4:
    printf("medium\n");
    break;

  default:
    printf("high\n");
}
```

- switch is applied to integer-valued variables such as char, int
- note the case and default keywords

Control Flow VI

- the `break` keyword causes to exit from the switch immediately without checking all the following cases
- `break` could also be used inside a `for`, `while` or a `do-while` loop and it lets you exit from the innermost enclosing loop immediately:

```
#define MM 5

// should only print MM / 2 = 2
for (ii = 0; ii < MM; ++ ii) {
    if (ii == MM / 2)
        break;
}
printf("%d\n\n", ii);

// a while loop version of the above for loop
ii = -1;
while (ii < MM) {
    ++ii;
    if (ii == MM / 2)
        break;
}
printf("%d\n\n", ii);
```

Control Flow VII

- the `continue` keyword when used inside a `for`, `while` or a `do-while` loop causes the next iteration of the innermost enclosing loop to begin immediately (*note it doesn't apply to switch*):

```
#define MM 5

// should print 0 through MM - 1 but MM / 2 = 2
for (ii = 0; ii < MM; ++ ii) {
    if (ii == MM / 2)
        continue;
    printf("%d\n", ii);
}

// a do-while loop version of the above for loop
ii = -1;
do {
    ++ii;
    if (ii == MM / 2)
        continue;
    printf("%d\n", ii);
} while (ii < MM - 1);
```

Code Files

prog3.c