Today’s Goals

• Short-hands, prefix and postfix
• for loops
• Arrays
• Arrays and chars
• ctype.h
• stdlib.h

Shorthand

```c
int i = 1, end = 100, val = 0;
while (i <= end) {
    val += i;
    i += 1;
}
```

Shorthands: op=, ++i, i++

- +=, -=, *=, /=, %=
- Prefix form increments i’s value before it is referenced
  ```c
  i = 5;
  x = (i++) + 6;
  ```
- Postfix form increments i’s value after it is referenced
  ```c
  i = 5;
  x = (i++) + 6;
  ```

for Loops

```c
int i = 1, end = 100, val = 0;
while (i <= end) {
    val += i;
    i += 1;
}
```

```c
int i, end = 100, val = 0;
for (i = 1; i <= end; i++) {
    val += i;
}
```

for Loop

- Pattern
  ```c
  for ( init; condition; update ) {
      body
  }
  ```
- Each section can be blank.
- Sequence: init | condition | update (cond fails)
**break** Statements

- Exit from a loop
- Typically used with an if statement
  (as in the previous page)

Example

```c
int i, val;
for(i=1, val=0; i<=100; i++) {
  if (val > 50)
    break;
  val += i;
}
```

**continue** Statements

- Continue to the beginning of a loop
  - I.e., the condition will be checked
- Typically used with an if statement

Example

```c
int i, val;
for (i=1, val=0; i<=100; i++) {
  if (i > 20 && i < 30)
    continue;
  val += i;
}
```

**Variations**

- No braces
- Omission of component(s)

```c
int i = 0;
for (; i < 10;) {
  printf("!n");
  i++;
}
```

**Nested for**

```c
int i, j, end = 10;
for (i = 1; i <= end; i++) {
  for (j = i; j <= i; j++) {
    printf("*");
  }
  printf("n");
}
```

```c
triangle.c
```
### Arrays

- To store a large number of data of homogenous type (e.g. `int` only)

- Schematic representation

```
    0 1 2 ... k-2 k-1
```

### Array Operations

- Declaration
  ```
  int a[5];
  ```

- Assignment
  ```
  a[0] = 1;
  ```

- Reference
  ```
  int y = a[0];
  ```

### Arrays and Characters

```c
int main() {
    int digits[10] = {0}, i; char c;
    while((c = getchar()) != EOF) {
        if (c >= '0' && c <= '9')
            digits[c-'0']++;
    }
    return 0;
}
```

### stdlib.h

- `void exit(int status);`
- Terminates a C program.
- Non-zero parameter values indicate program error to parent.
- A call to `exit(1)` is often used in conjunction with error detection.

### ctype.h

- C library containing a bunch of very useful character functions.
- These functions take an integer (not necessarily a `char`) and return 0 or 1.
- `int isdigit(int c);`
- `isalpha, isalnum, isspace, islower, isupper`
- `int tolower/toupper (int c);`

### Summary

- Be careful with prefix and postfix, especially postfix

- Loops are where a program spends most of its time. Learn to write efficient ones!

- Learn to use array and characters flexibly