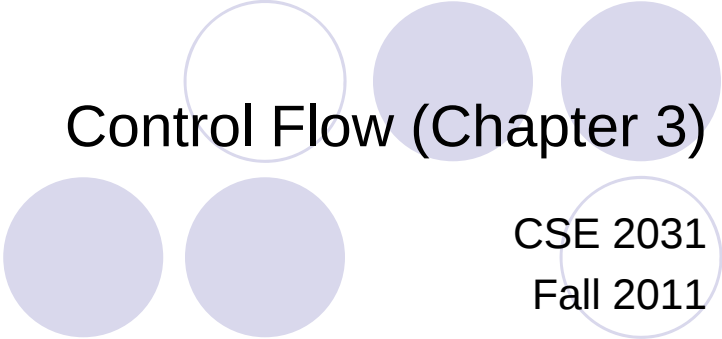


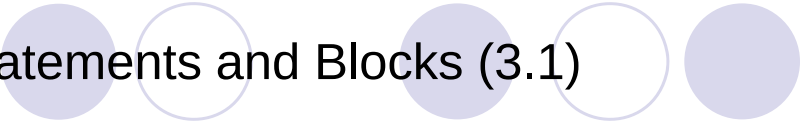


Control Flow (Chapter 3)

CSE 2031
Fall 2011

9/25/2011 12:08 PM

1



Statements and Blocks (3.1)

- Statement: followed by a semicolon.
- Block
 - enclosed between { and }
 - syntactically equivalent to a single statement
 - no semicolon after the right brace
- Variables can be declared inside *any* block.

2

Control Flow Statements

- Similar to Java
- **if - else**
- **else - if**
- **switch**
- **while**
- **for**
- **do - while**
- **break**
- **break**
- **continue**
- **goto**
- labels

3

if - else

```
if (n > 0)
    if (a > b)
        z = a;
    else
        z = b;
```

```
if (n > 0) {
    if (a > b)
        z = a;
}
else
    z = b;
```

4

if – else – if

```
int binary_search( int x, int v[], int n ) {
    int low, high, mid;
    low = 0;
    high = n - 1;
    while (low <= high) {
        mid = (low + high)/2;
        if (x < v[mid])
            high = mid + 1;
        else if (x > v[mid])
            low = mid + 1;
        else /* found match */
            return mid;
    }
    return -1; /* no match */
}
```

5

switch

```
while ((c = getchar()) != EOF) {
    switch (c) {
        case '0': case '1': case '2': case '3': case '4':
        case '5': case '6': case '7': case '8': case '9':
            ndigit[c-'0']++;
            break;
        case ' ':
        case '\n':
        case '\t':
            nwhite++;
            break;
        default:
            nother++;
            break;
    }
}
```

6

while and for Loops

```
while ((c = getchar()) == ' ' || c == '\n'
      || c == '\t')
    ; /* skip white space characters */
```

```
for (i = 0; i < n; i++)
    ...
```

7

do – while

```
do {
    s[i++] = n % 10 + '0';
} while ((n /= 10) > 0);
```

Note: the above curly brackets are not necessary. They just make the code more readable.

8

continue

Skip negative elements; increment non-negative elements.

```
for (i = 0; i < n; i++) {  
    if (a[i] < 0)      /* skip negative */  
        continue;  
    a[i]++; /* increment non-negative */  
}
```

9

break

Return the index of the first negative element.

```
...  
for (i = 0; i < n; i++)  
    if (a[i] < 0) /* 1st negative element */  
        break;  
if (i < n)  
    return i;  
...
```

10

goto and Labels

Determine whether arrays a and b have an element in common.

```
for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
        if (a[i] == b[j])
            goto found;
/* didn't find any common element */
...
found:
/* got one: a[i] == b[j] */
...
```

11

Notes

- Code that relies on **goto** statements is generally harder to understand and to maintain. So **goto** statements should be used rarely, if at all.
- **break** and **continue** should be used only when necessary.

12



Next time ...

- Arrays and pointers (chapter 5, C book)