Control Flow

Stanley Yao
Computer Science Department
University of Arizona

Conditional Branch

```
if (bool_expression)
    statement
```

- Example1:

```
if (i > 2)
   printf(“(TRUE) i is larger than 2\n”);
```

- Example 2:

```
if (i > 2) {
   printf(“(FALSE) I is larger than 2\n”);
   Printf(“this is the second statement\n”);
}
```

Conditional Branch (cont.)

- Bool expression .vs. integer value
  - 0: FALSE
  - Non-zero: TRUE
- Example 1:

```
if (5)
   printf(“TRUE”);
```

- Example 2:

```
printf(“%d”, (2==2)+5);
```

Conditional Branch (cont.)

```
if (bool_expression)
    statement_1
else
    statement_2
```

- Example1:

```
if (i > 2)
   printf(“(TRUE) i is larger than 2\n”);
else
   printf(“(FALSE) I is not larger than 2\n”);
```

- Example 2:

```
if (i > 2) {
   printf(“(TRUE) I is larger than 2\n”);
   Printf(“this is the second statement\n”);
} else {
   printf(“(FALSE) I is not larger than 2\n”);
   Printf(“one more statement\n”);
}
```

Multi-way Branch

```
if (bool_expression_1)
      statement_1
else if (bool_expression_2)
      statement_2
… … …
else if (bool_expression_n)
      statement_n
```

- Compare with switch-case
- Example:

```
if (i>=90)
   printf(“A”);
else if (i>=80)
   printf(“B”);
else if (i>=70)
   printf(“C”);
else
   printf(“D”);
```

if-else ambiguity

```
if (a > 0)
   if (b == 0)
       n = 1;
   else
       n = 2;
```

```
if (a > 0)
    if (b == 0)
        n = 1;
else
    n = 2;
```

?

Bool Expression

- Arithmetic operators
- Relational operators
  - >, >=, <, <=
  - ==, !=
- Logical operators
  - &&
  - ||

According to precedence

- (20 <(5+2)) ||((4>2)&& (3==3))
- Explicit precedence by "( )" is recommended!
- Bool Expression Short-circuit

Conditional Expression

```
if (a > b) {
        max = a;
} else {
        max = b;
}
```

```
max = (a > b) ? a : b;
```

switch-case

```
switch (expression){
  case const_1:
    // body
  case const_2:
    // body
  … … … …
  case const_n:
    // body
  default:
    // body
}
```

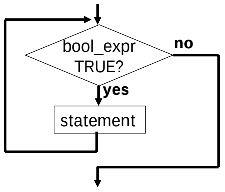
- Only can match integer constants
- "default" is optional
- Fall through
- "break"

switch-case (cont.)

```
switch (n) {
   case 4: printf("*");
   case 3: printf("*");
   case 2: printf("*");
   case 1: printf("*");
             break;
   case 5: printf("Full score!");
             break;
   Default:
             printf("Invalid score\n");
}
```

While Loop

```
while (bool_expr)
       statement
```

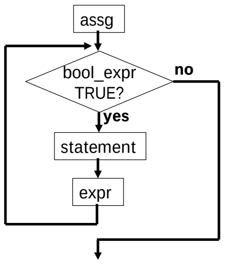


For Loop

```
for (assg; bool_expr; expr)
       statement
```

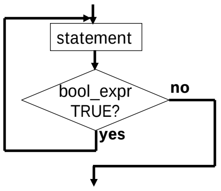
==

```
assg;
while (bool_expr) {
    statements;
    expr;
}
```



Do-while Loop

```
do
      statement
while (bool_expr);
```

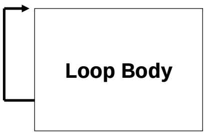


break

- switch-case
  - Jump out of the whole switch-case
  - Avoid fall through
- for, while, do-while Loop
  - Jump out of the whole loop
  - Terminate the loop
  - Execute the first statement right after the loop

continue

- for, while, do-while Loop
  - Doesn't jump out of the loop
  - Jump to the beginning of the loop
  - Terminate the current iteration
  - Begin to execute the next iteration



goto

- goto label_name;
- label_name:

```
goto next;
……
next:
……
```

- Normal usage:
  - The function has multiple error exits, but there exiting cleanup is the same
  - The function has multiple returns, but there return cleanup is the same

goto (cont.)

```
R_LOCK();
if ((ret=fun1(1,2)) != ERR_OK)
        goto err;
… …
if ((ret=fun2(a, b, c)) != ERR_OK)
        goto err;
… …
if (a>b)
        goto done;
… …
if ((ret=fun3()) != ERR_OK)
        goto err;
… …
err:    gErrCode = ret;
        cleanup();
done:   R_UNLOCK();
        return ret;
```

Special Usage & Example

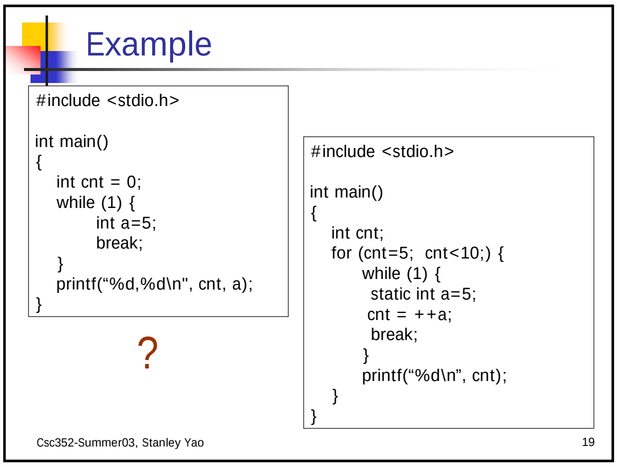
- for (;;) { … }
- while (1) { … }
- while (p) { … } // p is a pointer

```
int x = 5;
while (x > 0);
    x --;
printf("%d", x);
```

?

```
#include <stdio.h>

int main()
{
    int cnt = 0;
    char c;
    while ((c = getchar()) != EOF)
        cnt ++;
    printf("\nTotal = %d\n", cnt);
}
```



Example

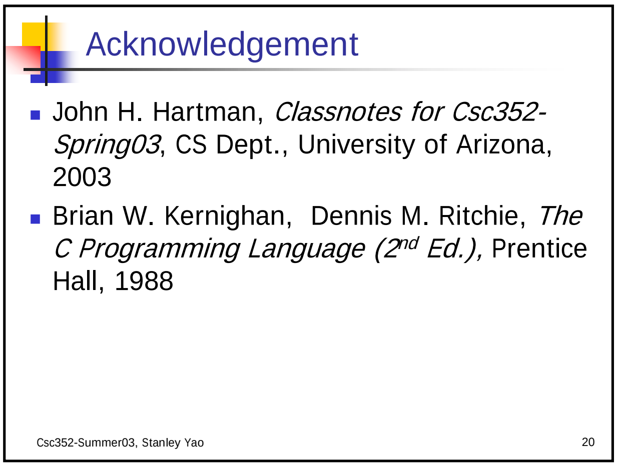
```
#include <stdio.h>

int main()
{
    int cnt = 0;
    while (1) {
           int a=5;
           break;
    }
    printf(“%d,%d\n", cnt, a);
}
```

?

```
#include <stdio.h>

int main()
{
    int cnt;
    for (cnt=5;  cnt<10;) {
          while (1) {
            static int a=5;
           cnt = ++a;
            break;
          }
          printf(“%d\n”, cnt);
    }
}
```



Acknowledgement

- John H. Hartman, *Classnotes for Csc352-Spring03*, CS Dept., University of Arizona, 2003
- Brian W. Kernighan, Dennis M. Ritchie, *The C Programming Language (2nd Ed.),* Prentice Hall, 1988