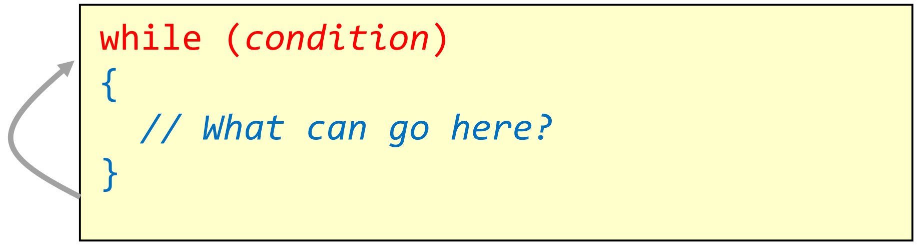
Unit 11

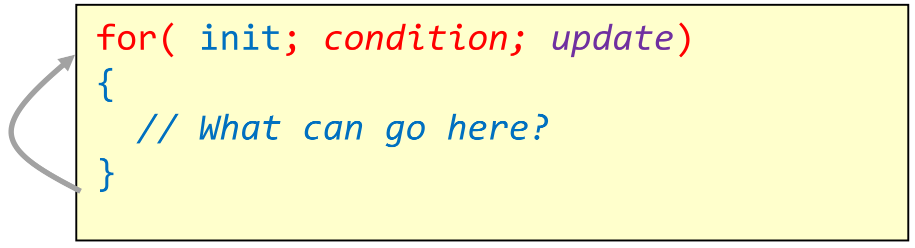
Nested Loops

What Can Go Inside?

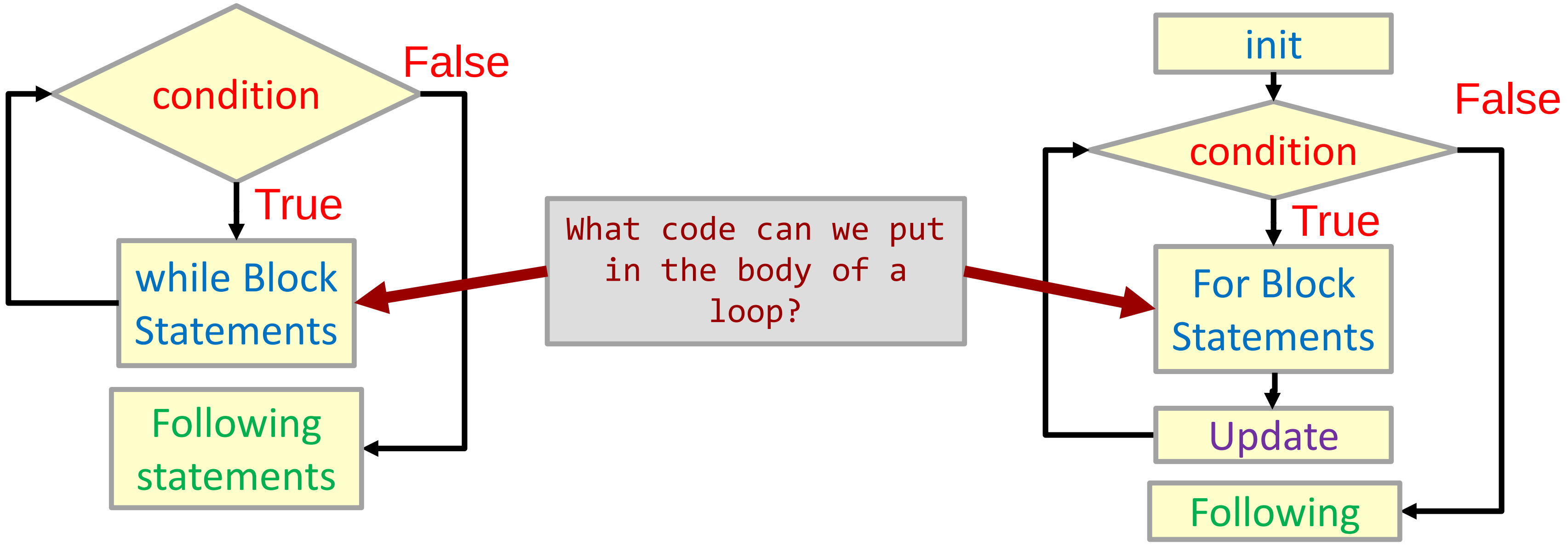
- What kind of code can we put in the body of a loop?
- ANYTHING…even other loops



```
while (condition)
{
  // What can go here?
}
```



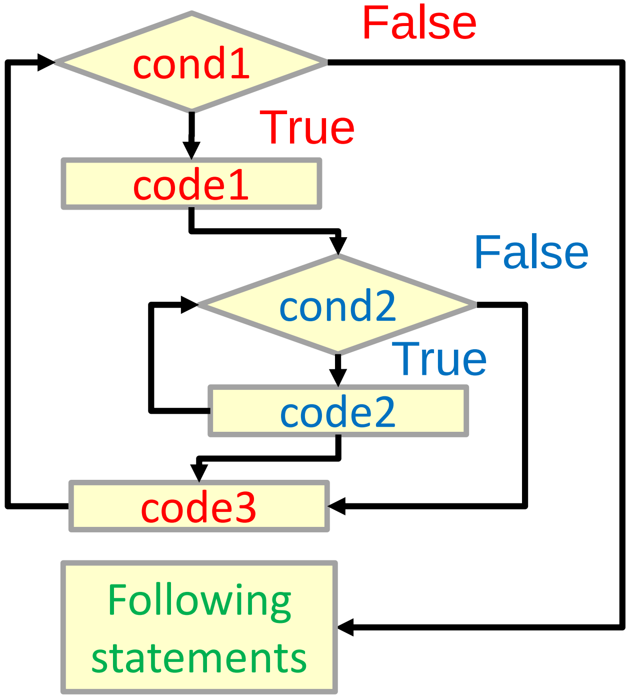
```
for( init; condition; update)
{
  // What can go here?
}
```



Nested Loops

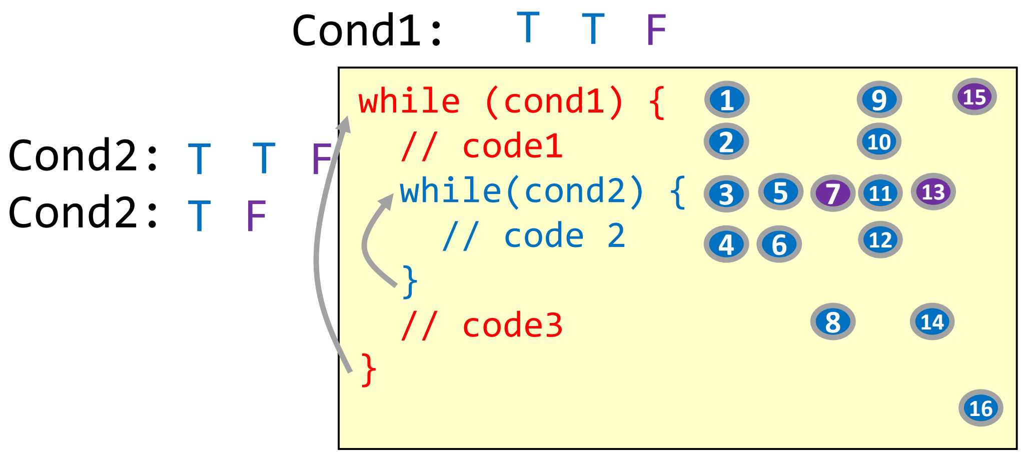
- Loops can contain other loops in their body

```
while (cond1) {
  // code1
  while(cond2) {
    // code 2
  }
  // code3
}
```

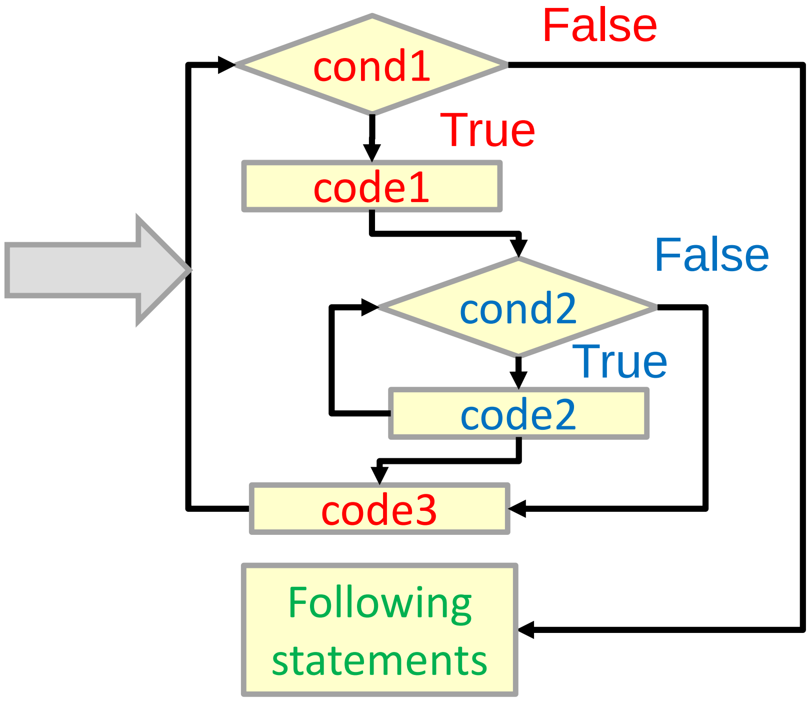


Nested Loop Sequencing

- **Key Idea**: The inner loop runs in its entirety for each iteration of the outer loop



```
while (cond1) {
  // code1
  while(cond2) {
     // code 2
  }
  // code3
}
```



Nested Loops Example 1

- When you write loops consider what the body of each loop means in an abstract sense
  - The body of the outer loop represents 1 game (and we repeat that over and over)
  - The body of the inner loop represents 1 turn (and we repeat turn after turn)

1 game

1 turn

```
int main()
{
  int secret, guess;
  char again = 'y';
  // outer loop
  while(again == 'y')
  {  // Choose secret num. 0-19
     secret = rand() % 20;
     guess = -1;
     // inner loop
     while(guess != secret)
     {
       cout << "Enter guess: ";
       cin >> guess;
     }
     cout << "Win!" << endl;
     cout << "Play again (y/n): ";
     cin >> again;
  }
  return 0;
}
```

Nested Loops Example 2

- **Key idea**: Perform all iterations of the inner loop before starting the next iteration of the outer loop
  - Said another way: The inner loop executes completely for each single iteration of the outer loop
- Trace through the execution of this code and show what will be printed

```
int main()
{
  for(int i=0; i < 2; i++){
    for(int j=0; j < 3; j++){
      cout << i << " " << j << endl;
    }
  }
}
```

<u>i</u> <u>j</u>

Nested Loops Example 3

- Trace through the execution of this code and show what will be printed if the user types in: 8 4 7 6

```
int main()
{
  int x = 0;
  cin >> x;
  while( x%2 == 0 ){
    for(int i=x; i >= 0; i -= 2){
      cout << i << " ";
    }
    cout << endl;
    cin >> x;
  }
  cout << "Done" << endl;
  return 0;
}
```

Program Output:

break Statement with Nested Loops

- break will only exit the inner-most loop, not all the nested loops.
- This can be exactly what you want in some cases
- In other cases, you may want to break out of all loops, but realize a single 'break' statement cannot do that.
  - Instead must change a variable so that the outer loop condition will fail

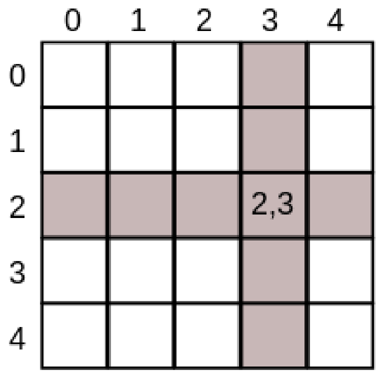
```
char again = 'y';
while(again == 'y' )
{
  /* Give the user 10 turns
        but stop if guess right */

  int i, guess, secretNum = /*..*/
  for(i=0; i < 10; i++)
  {
    cin >> guess;
    if(guess == secretNum){
      break;
    }
  }
  if( i == 10 )
    cout << "You lose!" << endl;
  else
    cout << "You win!" << endl;

  cin >> again;
}
```

Tips

- Nested loops often help us represent and process multi-dimensional data
  - 2 loops allow us to process data that corresponds to 2 dimension (i.e. rows/columns)
  - 3 loops allow us to process data that corresponds to 3 dimensions (i.e. rows/columns/planes)



More Practice

- cpp/nestedloops/rectangle
- cpp/nestedloops/flag
- cpp/nestedloops/etox-range
- cpp/nestedloops/sphere

SOLUTIONS

Nested Loops Example 2

- Trace through the execution of this code and show what will be printed

```
int main()
{
  for(int i=0; i < 2; i++){
    for(int j=0; j < 3; j++){
      cout << i << " " << j << endl;
    }
  }
}
```

i	j
0	0
	1
	2
1	0
	1
	2

Program Output:

```
0 0
0 1
0 2
1 0
1 1
1 2
```

Nested Loops Example 3

- Trace through the execution of this code and show what will be printed if the user types in: 8 4 7 6

```
int main()
{
  int x = 0;
  cin >> x;
  while(x%2 == 0){
    for(int i=x; i >= 0; i -= 2){
      cout << i << " ";
    }
    cout << endl;
    cin >> x;
  }
  cout << "Done" << endl;
  return 0;
}
```

Program Output:

```
8 6 4 2 0
4 2 0

Done
```