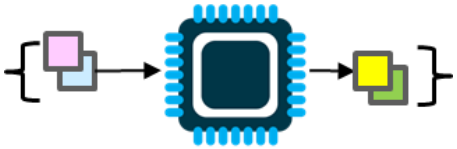
Unit 3a – Nested Loop Tracing

Mark Redekopp

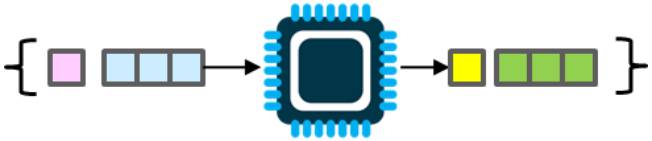
© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Unit 3

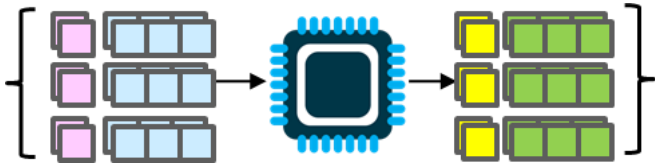
- **Unit 1**: Scalar processing
  - aka IPO=Input-Process-Output Programs



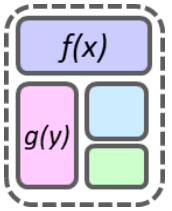
- **Unit 2**: Linear (1D) Processing



- **Unit 3**: Multidimensional Processing



- **Unit 4**: Divide & Conquer (Functional Decomposition)

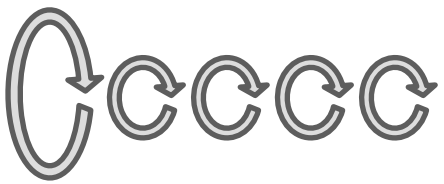


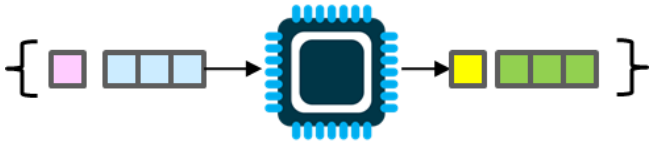
© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

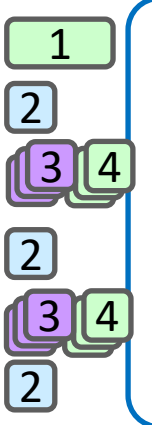
This Photo by Unknown Author is licensed under CC BY-NC

Multidimensional Processing

- Most non-trivial programs require "multidimensional" processing.
- The distinguishing feature is the use of **NESTED** loops
  - Each nested loop represents one dimension
  - A representative example might be an outer loop to get multiple inputs and an inner loop to process each input
- We will likely still keep our general structure but repeat a subset in each nested loop:
  - Prompt
  - Input
  - Process
  - Output








```
Enter numbers to factor (end with -1)
9
1 3 9
12
1 2 3 4 6 12
-1
```


```
Enter rectangle's base & height:
3 5
*****
*****
*****
```

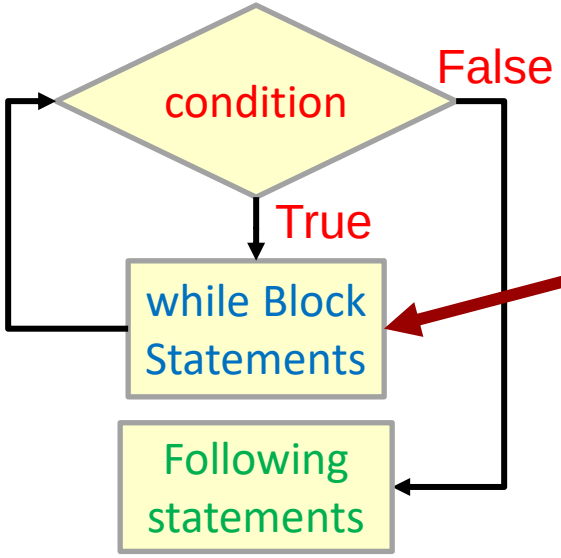
© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Loops Inside Loops

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

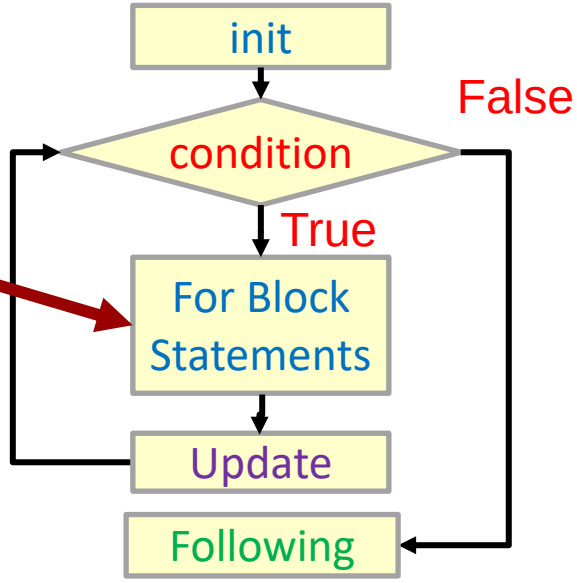
```
while (condition)
{
  // What can go here?
}
```

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



What code can we put in the body of a loop?

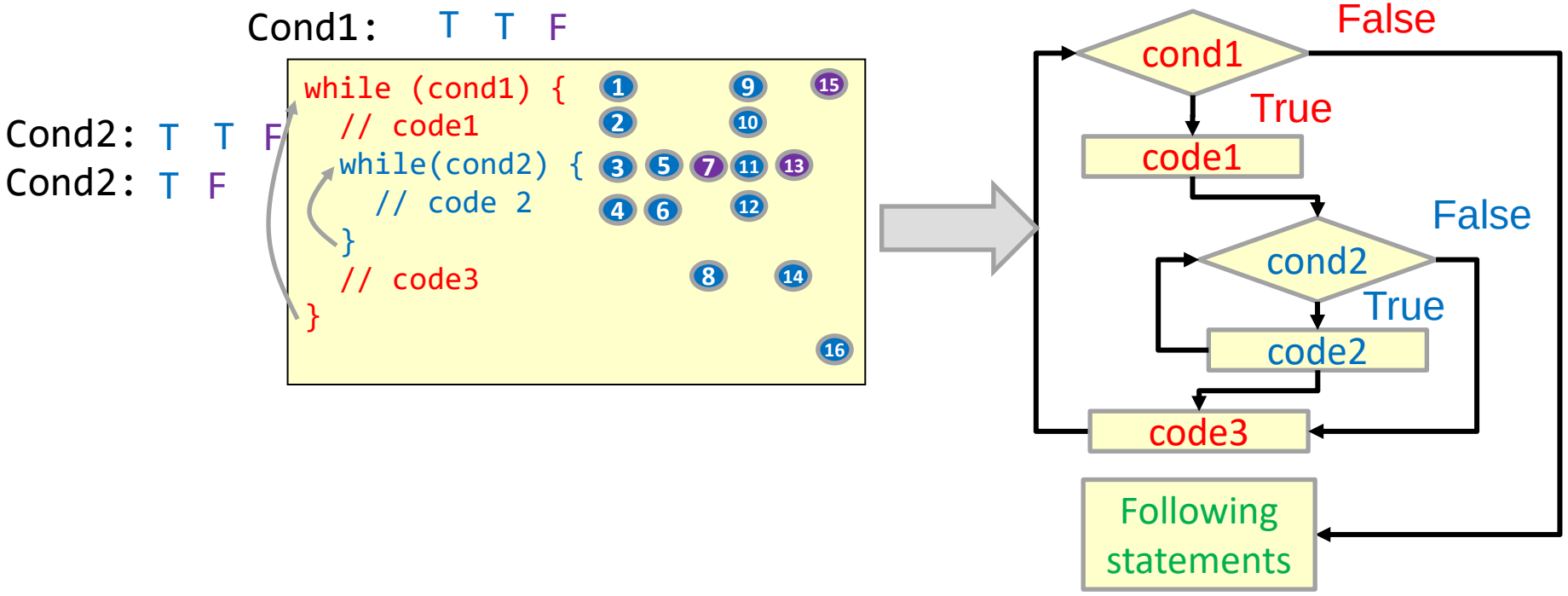
Another loop!



© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Nested Loop Sequencing

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



© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

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)

```
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;
}
```

1 game

1 turn

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

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
- **Exercise**: rectangle

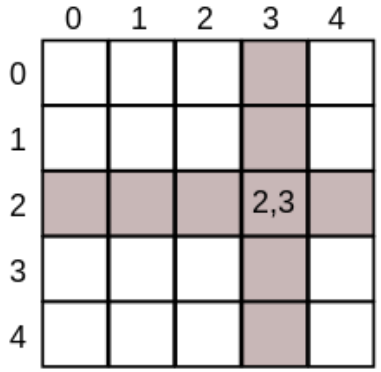
```
int main()
{
  for(int i=0; i < 2; i++){
    for(int j=0; j < 3; j++){
      cout << i << " " << j << endl;
    }
  }
}
```

i j

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Visualizing Nested Loops

- 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)



© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Sequential vs. Nested Loops

- **Sequential loops** are run one after the other
  - Each loops runs to completion before the next starts
- **Nested loops** runs the inner loop to completion for each iteration of the outer loop

Program Output:

```
0 1 2 3       // Sequential Loops
3 2 1 0

              // Nested Loops
_____________
_____________
_____________
_____________
```

```
int main()
{
  // Sequential Loops
  for(int i=0; i < 4; i++){
     cout << i << " ";
  }
  cout << endl;
  for(int k=3; k >= 0; k--){
     cout << k << " ";
  }
  cout << endl << endl;

  // Nested Loops
  for(int i=0; i < 4; i++){
     cout << i << " ";
     for(int k=3; k >= 0; k--){
        cout << k << " ";
     }
     cout << endl;
  }
  return 0;
}
```

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

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

```
int main() {
  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;
  }
  return 0;
}
```

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Nested Loops Example 3

```
int main()
{
  int num;
  cout << "Enter numbers to factor (end with -1)" << endl;
  cin >> num;   // get first num
  while(num != -1){
    // find all factors of num, one at a time
    for(int factor = 1; factor <= num; factor++){
      if(num % factor == 0) {
        cout << factor << " ";
      }
    }
    cout << endl;
    cin >> num; // get next num
  }
  return 0;
}
```

Program Output:

```
Enter numbers to factor (end with -1)
9
1 3 9
12
1 2 3 4 6 12
-1
```

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Nested Loops Example 4

- 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:

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

SOLUTIONS

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

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
```

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Nested vs. Sequential Loops

- **Sequential loops** are run one after the other
  - Each loops runs to completion before the next starts
- **Nested loops** runs the inner loop to completion for each iteration of the outer loop

```
int main()
{
  // Sequential Loops
  for(int i=0; i < 4; i++){
     cout << i << " ";
  }
  cout << endl;
  for(int k=3; k >= 0; k--){
     cout << k << " ";
  }
  cout << endl << endl;

  // Nested Loops
  for(int i=0; i < 4; i++){
     cout << i << " ";
     for(int k=3; k >= 0; k--){
        cout << k << " ";
     }
     cout << endl;
  }
  return 0;
}
```

Program Output:

```
0 1 2 3
3 2 1 0

0 3 2 1 0
1 3 2 1 0
2 3 2 1 0
3 3 2 1 0
```

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.

Nested Loops Example 4

- 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
```

© 2023 by Mark Redekopp. This content is protected and may not be shared, uploaded, or distributed.