Cal Poly | **CPE 101:** Fundamentals of Computer Science I | Alexander Dekhtyar

C Programs: Loops

Loops

Loops are control structures that tell CPU to repeat a sequence of actions a number of times.

C facilitates **repetition** of the same set of instructions in the program via three types of loops.

counting loops. Used when the programmer knows exactly how many repetitions of the loop is needed.

precondition loops. Used when the exact number of repetitions is not known. A loop termination condition is checked **before** each loop iteration (thus it is possible for the loop to not run at all).

postcondition loops. Used when the exact number of repetitions is not known. A loop termination condition is checked **after** each loop iteration (therefore, the loop will repeat at least once).

The for loop

Syntax.

```
for ( <initExpr>;
      <repeatExpr>;
      <updateExpr>)
    <statementBlock>
```

	`<initExpr>`:	initialization expression. Usually, an assignment
	`<repeatExpr>`:	loop repetition expression. Usually, a comparison
Here,	`<updateExpr>`:	expression that updates program state before every iteration. Usually an assignment and an increment/decrement
	`<statementBlock>`:	statement (simple or compound) to be executed

Example.

```
sum = 0;
```

```
for (i = 0; i<=10; i++) {
   sum = sum+i;
}
```

Semantics The components of the loop are executed in the following order:

```
<initExpr>        /* executes before the first iteration of the loop */
<statementBlock>
<updateExpr>      /* executes before every iteration of the loop     */
                  /* except for the first iteration                  */

if (<repeatExpr>) {
   <statementBlock>
   <updateExpr>
   if (<repeatExpr) {
      <statementBlock>
      <updateExpr>
      if (<repeatExpr>) {
         ...
      }
   }
 }
```

That is, `<statementBlock>` and `<updateExpr>` execute **until** for as long as `<repeatExpr>` **evaluates to true**.

Example.

```
for (i=1; i<=10;i++) {
   scanf("%d",&x);
   scanf("%d",&y);
   r = x*x + y*y;
   printf("%d", r);
}
```

Nested Loops

`For` loops can nest inside each other.

The following loop outputs the distance to the (0,0) for each point inside the 11x11 square.

```
int i,j;
float dist;
for(i=0; i<=10;i++) {
   for(j=0; j<=10;j++) {
      dist = sqrt(i*i+j*j);
      printf("(%d,%d) = %f\n",i,j,dist);
   }
}
```