UNIVERSITY of LOUISVILLE
dare to be great

C Chapter 4: Looping Structures

CECS130
Introduction to Programming Languages
Dr. Roman V. Yampolskiy

Loops

- Purpose of the loop is to repeat part of the algorithm
- Examples of situations requiring loops:
  - Displaying a menu
  - Playing a game until the game is over
  - Keeping the air conditioning on until desired temperature is met
  - Asking user for data until desired data is obtained
- Make it possible for your program to run forever (infinite loop)
  - Usually result of a programming error

Operators (again)

- Some operators are particularly useful with loops
- ++
- Can be used either as X++ or as ++X

```
#include <stdio.h>
main() {
    int x = 0; int y = 0;
    printf("\n The value of y is %d\n", y++);
    printf("\n The value of x is %d\n", ++x);
}
```

The value of y is 0

The value of x is 1

Operators (again)

- **--**
- Can be used as either prefix or as postfix

```
#include <stdio.h>
main() {
    int x = 0; int y = 0;
    printf("\n The value of y is %d\n", y--);
    printf("\n The value of x is %d\n", --x);
}
```

The value of y is 0

The value of x is -1

The while Loop

- The while statement is a general repetition statement with the following syntax:

```
while(expression)
{
    statement(s);
}
```

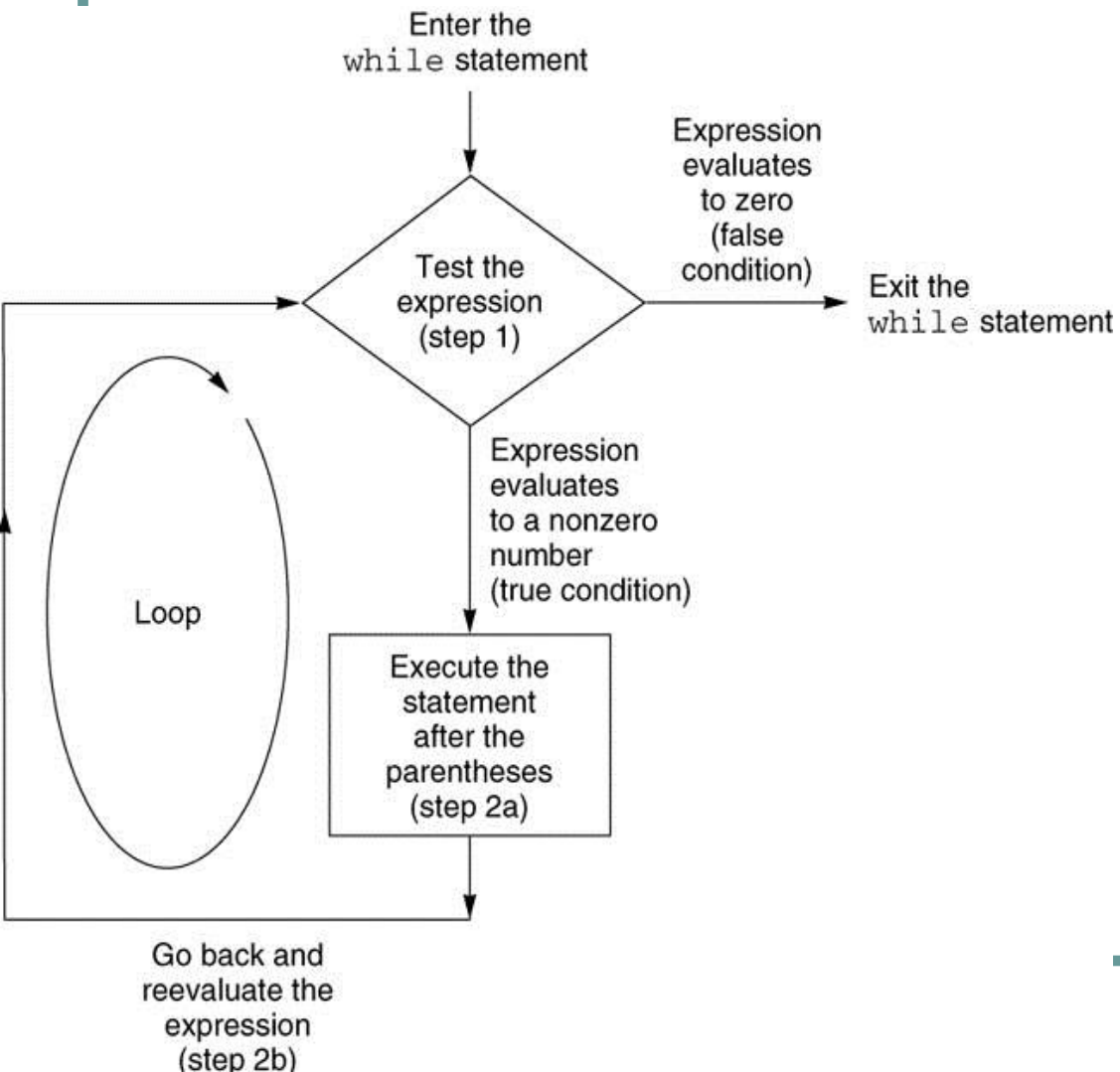
- The process used to execute this construct is as follows:
  1. Test the expression.
  2. If expression is not 0 (not false)

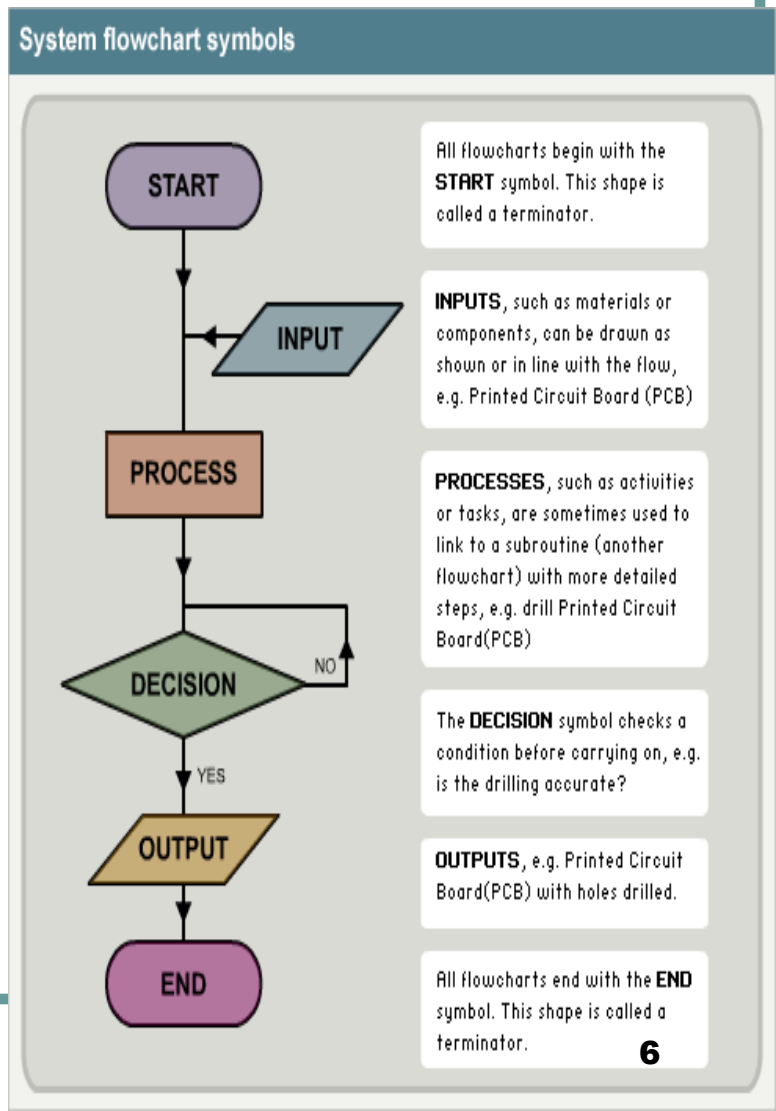
     execute the statements in the brackets and go back to step 1.

     else

     exit the while statement

The while Loop: Flowchart





The while Loop Example: Factorial

```
unsigned int counter = 5;
unsigned long factorial = 1;
while (counter > 0)
{
    factorial *= counter--;
                    //Multiply and decrement
                    // Same as:
                    // factorial = factorial * counter;
                    // counter = counter – 1;
 }
printf("%i", factorial);
```

The for Loop

- The “for” statement performs the same task as a while statement.
- The “for” statement has its built in counter that must be initialized, tested, and updated as part of the “for” structure.
- It typically looks as follows:

```
int counter;
for (counter = initialValue; counter <= bound; counter = counter + increment)
{
    Statemenet(s);
}
```

  - Where counter must be defined before the for statement.
  - Initial value is starting value, bound is the upper bound to be reached and increment is the amount used to increment the counter.
  - “for” statement may be used in both escalating loops or de-escalating loops.
  - “for” statement is customarily used <u>when the number of repetitions is known</u> but should not be the case.

The for Loop Example: Fibonacci

Sign in / create account

article | discussion | edit this page | history

Your continued donations keep Wikipedia running!

WIKIPEDIA
The Free Encyclopedia

navigation
- Main page
- Community portal
- Featured content
- Current events
- Recent changes
- Random article
- Help
- Contact Wikipedia
- Donations

search

Go | Search

toolbox
- What links here
- Related changes
- Upload file
- Special pages

Fibonacci number

From Wikipedia, the free encyclopedia

In mathematics, the **Fibonacci numbers** form a sequence defined by the following recurrence relation:

$$F(n) := \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases}$$

That is, after two starting values,each number is the sum of the two preceding numbers. The first Fibonacci numbers (sequence A000045 in OEIS), also denoted as F_n, for $n = 0, 1, \ldots$, are:

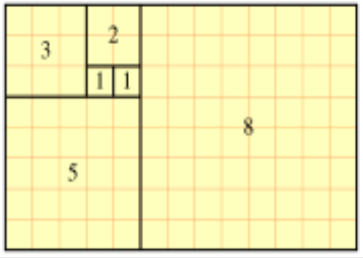
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025...

Sometimes this sequence is considered to start at $F_1 = 1$, but it is more common to include $F_0 = 0$.

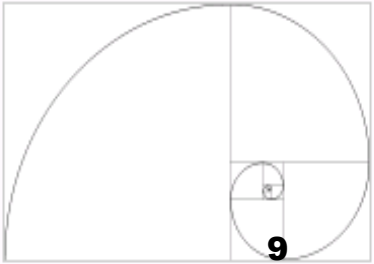
The Fibonacci numbers are named after Leonardo of Pisa, known as Fibonacci, although they had been described earlier in India.[1][2]

Contents [hide]

1 Origins



A tiling with squares whose sides are successive Fibonacci numbers in length



A Fibonacci spiral

The for Loop Example: Fibonacci

```
/* Print Fibonacci numbers and quotients. */

#include <stdio.h>

#define   LIMIT   15

int main(void)
{
   long   f0 = 0, f1 = 1, n, temp;

   printf("%7s%19s%29s\n%7s%19s%29s\n%7s%19s%29s\n",    /* headings */
      "  ", "Fibonacci", "Fibonacci",
      " n", "   number", " quotient",
      "--", "---------", "---------");
   printf("%7d%19d\n%7d%19d\n", 0, 0, 1, 1);     /* first two cases */

   for (n = 2; n <= LIMIT; ++n)
   {
      temp = f1;
      f1 += f0;
      f0 = temp;
      printf("%7ld%19ld%29.16f\n", n, f1, (double) f1/f0);
   }
   system("PAUSE");
   return 0;
}
```

The for Loop Example: Fibonacci

```
                Fibonacci                    Fibonacci
     n             number                     quotient
    --          ---------                    ---------
     0                 0
     1                 1
     2                 1          1.0000000000000000
     3                 2          2.0000000000000000
     4                 3          1.5000000000000000
     5                 5          1.6666666666666667
     6                 8          1.6000000000000001
     7                13          1.6250000000000000
     8                21          1.6153846153846154
     9                34          1.6190476190476191
    10                55          1.6176470588235294
    11                89          1.6181818181818182
    12               144          1.6179775280898876
    13               233          1.6180555555555556
    14               377          1.6180257510729614
    15               610          1.6180371352785146
Press any key to continue . . .
```

The do-while Loop

- The “do… while” loop looks as follows:

```
do {
        statement(s);
} while (condition);
```

- for, and while are called top nested loops this means that the condition is evaluated first than the body of the loop is executed (possibly zero times)
- do … while is called bottom nested, i.e. the condition is evaluated after the loop. This means that the Loop will execute at least once.

The do…while Loop Example: Factorial

```
unsigned int counter = 5;
unsigned long factorial = 1;
do
{
    factorial *= counter--;      /*Multiply, then decrement.*/
} while (counter > 0);
printf("%lu\n", factorial);
```

Break Statement

- Used to manipulate loops and switch statements
- If a break statement is executed in a loop, the loop is terminated and program control returns to the first statement after the loop

```
#include <stdio.h>
main() {
int x;
for (x = 10; x > 5; x--) {
    if (x == 7)                    //loop statements are executed only 4 times
        break;
    printf(”\n%d\n”, x);
}
printf(”\n%d\n”, x);
```

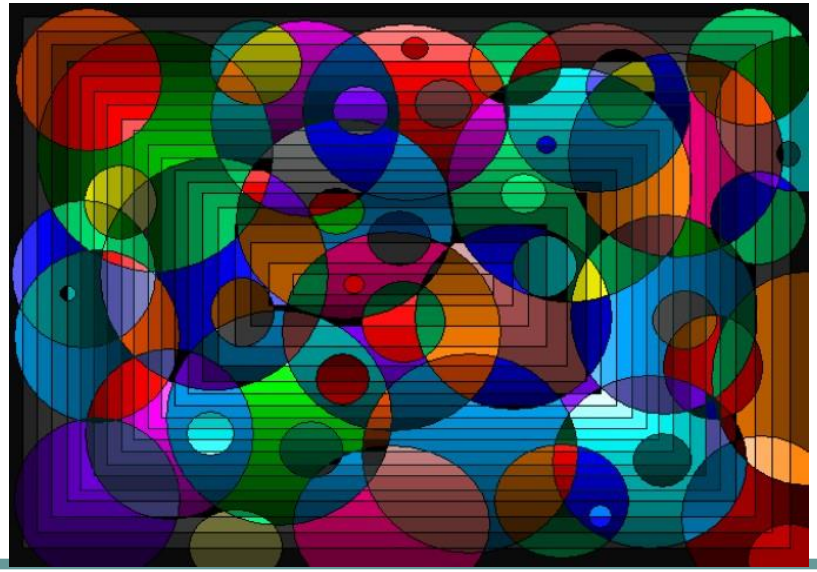
Continue Statement

- Used to manipulate loops
- If a continue statement is executed in a loop, any remaining statements in the loop body are passed over and the next iteration of the loop is commenced

```
#include <stdio.h>
main() {
int x;
for (x = 10; x > 5; x--) {
    if (x == 7)
        continue;                    //printing of 7 is skipped
    printf("\n%d\n", x);
}
```

Nested Loops

- Loops can be placed inside other loops
- Loops of different types (for, while, do while) can be mixed
- The number of times the deepest nested statement is executed is equal to the product of the number of iterations of the outside loops



Nested Loop – Example: Math Game

```c
#include <stdio.h>
int main(void) {
  int answer, i, chances, right;

  for(i = 1; i < 11; i++) {
    printf(" %d + %d = ?", i , i);
    scanf("%d", &answer);

    if(answer == i + i)
        printf("Right!\n");
    else {
      printf("Wrong.\n");
      printf("Try again.\n");

      right = 0;
      /* nested for */
      for(chances = 0; chances < 3 && !right; chances++) {
        printf(" %d + %d = ? ", i, i);
        scanf("%d", &answer);

        if(answer == i + i) {
          printf("Right!\n");
          right = 1;
        }
      }
      /* if answer still wrong, tell user */
      if(!right)
        printf("The answer is %d.\n", i + i);
    }
  }
  return 0;
}
```

```
 1 + 1 = ?2
Right!
 2 + 2 = ?4
Right!
 3 + 3 = ?6
Right!
 4 + 4 = ?3
Wrong.
Try again.
 4 + 4 = ? 4
 4 + 4 = ? 5
 4 + 4 = ? 6
The answer is 8.
 5 + 5 = ?10
Right!
 6 + 6 = ?12
Right!
 7 + 7 = ?14
Right!
 8 + 8 = ?16
Right!
 9 + 9 = ?18
Right!
 10 + 10 = ?20
Right!
```

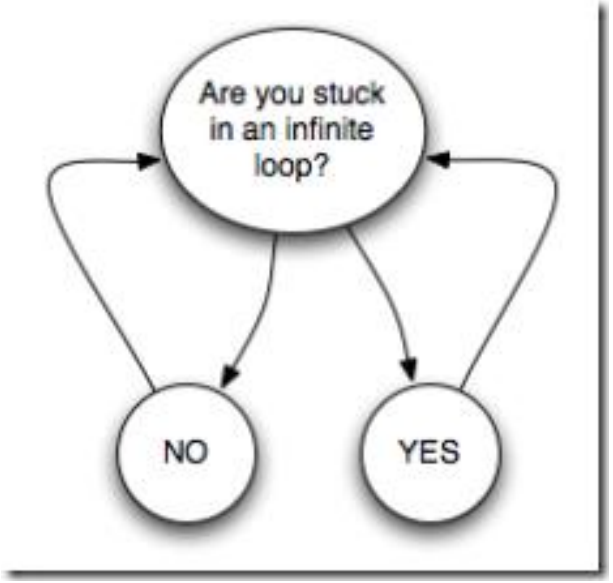
Converting for loop to while loop

```
for ( expr1; expr2; expr3 ){
        statement1;
}

expr1;
while (expr2) {
   statement1;
   expr3;
}
```

Infinite Loops

- Infinite loops are loops that never terminate.
- Example: "Lather, rinse, repeat" instructions on a shampoo bottle
- To get out, press Ctrl+Alt+Del, select your program and End Task



Infinite Loops: Example

```
unsigned int i;
for (i = 1; i > 0; i++)
{
    printf("%d", i);
}
```

More Examples: while loop

- Keeping a running sum

```
void main() {
    int sum = 0, number = 0;
    while( number != -1 ) {
        sum += number;
        printf( "The running sum is: %d\n", sum );
        printf( "Enter a positive integer (-1: to exit):");
        scanf( "%d", &number );
    }
}
```

More Examples: while loop

- Another way to do it

```
void main() {
    int sum = 0, number;
    while( 1 ) {
        printf( "The running sum is: %d\n", sum );
        printf( "Enter a positive integer (-1: to exit):" );
        scanf( "%d", &number );
        if( number == -1 ) break;
        sum += number;
    }
}
```

More Examples: for loop

- Adding the first ten positive even numbers (2, 4, 6, ..., 20).

```
void main() {
    int i, sum = 0;
    for( i = 1; i <= 10; i++ )
        sum += 2 * i;
    printf( "The sum is %d\n", sum );
}
```

More Examples: for loop

- Another way to do it

```
void main() {
    int i, sum = 0;
    for( i = 2; i <= 20; i += 2 )
        sum += i;
    printf( "The sum is %d\n", sum );
}
```

More Examples: do…while

```
/* Example of
   using the do/while repetition structure */
#include <stdio.h>

int main()
{
   int counter = 1;

   do {
      printf( "%d  ", counter );
   } while ( ++counter <= 10 );

   return 0;
}
```

More Examples: do…while

```
#include<stdio.h>
int main(void) {

    const int SECRET_CODE = 15;
    int code;

    do {
        printf("Type the secret code number to enter.\n");
        scanf("%d", &code);
    } while (code != SECRET_CODE);
    printf("Well done , you can now enter\n");
return 0;
}
```

System Calls

- C provides ability to execute operating system commands
- UNIX (ls, man, ps, etc.) or Windows (pause, cls, cd, etc.)
- Example:

```
#include <stdio.h>
main() {
    printf (“Print some text\n") ;
    system("cls");              //clear screen
    system("pause");            // wait
}
```

The End!