Programming Language Lab

Bibhas Adhikari

IIT Kharagpur

September 22, 2020

Loop structure

While loop

- *while* loop structure is used to create iteration (loops) in a program

- press Ctrl+C to exit an infinite loop
- press Ctrl+Alt+Del

```
main()
{
int x = 0;
while (x < 10){
printf(``The value of x is %d\n",x);
x++;
}
}
```

Loop structure

***do While* loop**

- *while do* loop structure is used to create iteration (loops) in a program

- Why use the do while loop instead of the while loop?

```
main()
{
int x = 10;
do {
printf("This printf statement is executed at least once
\n");
x++;
} while (x < 10);
}
```

Loop structure

for loop

- *for* loop structure is used to create iteration (loops) in a program, and it is more common than the previous loops
- A single for loop statement contains three separate expressions
  - Variable initialization
  - Conditional expression
  - Increment/decrement
- *for* loop can be used the number of times it would be executed is not known

```
main()
{
int x;
for (x = 10; x > 5; x − −)
printf(”The value of x is %d\n”, x);
}
```

Loop structure

break and *continue* statements

- *break* and *continue* are used to manipulate and control the program flow in loops
    - When a *break* statement is executed in a loop, the loop is terminated and program control returns to the next statement following the end of the loop
    - when a *continue* executed in a loop, any remaining statements in the loop are passed over and the next iteration of the loop is sought.

```
main()
{
int x;
for (x = 10; x > 5; x--){
if (x == 7)
break;
}
printf("\n%d\n", x);
}
```

Loop structure

```
main()
{
int x;
for (x = 10; x > 5; x--){
if (x == 7)
continue;
printf("\n%d\n", x);
}
}
```

Structured programming

The most relevant structured programming concepts are the following:

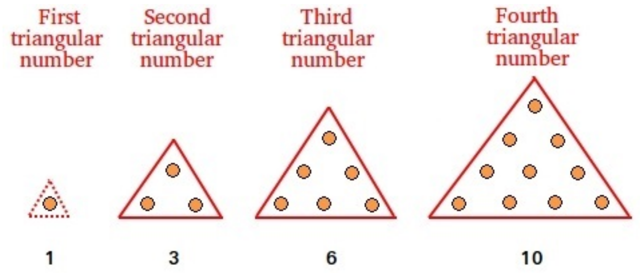
- Top-down design
- Code reusability

function

Structured programming enables programmers to break complex systems into manageable components, called *functions* in C

- function prototype declaration
- formal parameter name
- automatic local variables

Structured programming: example of a function



```
calculateTriangularNumber (int n)
{
int i, triangularNumber = 0;
for (i = 1; i <= n; ++ i)
triangularNumber += i;
printf ("Triangular number %i is %i\n", n, triangularNumber);
}
```

Structured programming: example of a function

```
main ()
{
calculateTriangularNumber (10);
calculateTriangularNumber (20);
calculateTriangularNumber (50);
}
```

```
gcd (int p, int q)
{
int temp;
printf ("The gcd of %i and %i is ", p, q);
while (q! = 0){
temp = p % q;
p = q;
q = temp;
}
printf (“%i\n”, p);
}
```

Example of some functions in C

Library Name	Function Name	Description
Standard input/output	*scanf()*	Reads data from the keyboard
Standard input/output	*printf()*	Prints data to the computer monitor
Character handling	*isdigit()*	Tests for decimal digit characters
Character handling	*islower()*	Tests for lowercase letters
Character handling	*isupper()*	Tests for uppercase letters
Character handling	*tolower()*	Converts character to lowercase
Character handling	*toupper()*	Converts character to uppercase
Mathematics	*exp()*	Computes the exponential
Mathematics	*pow()*	Computes a number raised to a power
Mathematics	*sqrt()*	Computes the square root