Control Flow

Arash Rafiey

August 29, 2017

Statements and Blocks

C **program** is a sequence of declarations and statements.

Declaration declares a variable/function where as **definition** both declares and defines it.

Statement can be simple (assignment, function call, return) or compound (block, if-else, switch, loops).

Statement terminator in C is a semicolon ;

Braces { and } groups declarations and statements and into logical units called **blocks**.

For example, braces surround the statements of a **function** or a **switch**.

Control Flow

Control flow defines the order in which instructions are run.

Control flow statement "makes" a choice which path to follow:

1. If-else statement
2. switch statement
3. Loop: for, do, while
4. Unconditional jump: goto
5. Function call
6. return;
7. Program halt

Although **branching** make programs slow, almost every program contains them.

If-else statement

If-else statement chooses a branch depending on the condition.

Syntax:

```
if (condition)
  consequent
else //The else part is optional.
  alternative
```

Execution order:

1. First, the "**condition**" is evaluated.
2. If it is true, the "**consequent**" is executed.
3. Otherwise, if there is else block, the "**alternative**" is executed.

If-else statement ambiguity

Since else block is optional, there might be an **ambiguity**:

```
if (a > b)
   if (c > a)
      m = c;
   else
      m = a;
```

Note. **C compiler** ignores **white space** (tab, space, new line).
Ambiguity resolution rule: else goes to the innermost if.

To change the order, use **braces**:

```
if (a > b) {
   if (c > a) //Second if hasn't else part.
      m = c;
}
else
   m = a;
```

Else-if

Syntax:

```
if (condition0)
  statement0
else if (condition1)
  statement1
else if (condition2)
  statement2
else //The last else part is optional.
  statement3
```

The conditions are evaluated **until some condition is true**.

In such case, the statement below it is executed; chain terminates.

The last else part is run when **none** of the conditions is satisfied.

Else-if example

Compare two numbers example :

Input: two integers.

Output: -1, 0, or 1 depending on whether the first number is less than, equal to, or greater than the second number.

```
int main () {
int a, int b
   if (a < b)
     return -1;
   else if (a > b)
     return 1;
   else
     return 0;
}
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
    if (b < c)
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
    if (b < c)
       printf("ordering %d,%d,%d \n", a, b, c);
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
    if (b < c)
       printf("ordering %d,%d,%d \n", a, b, c);
    else if (c < a)
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
   if (b < c)
      printf("ordering %d,%d,%d \n", a, b, c);
   else if (c < a)
         printf("ordering %d,%d,%d \n", c, a, b);
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
   if (b < c)
      printf("ordering %d,%d,%d \n", a, b, c);
   else if (c < a)
         printf("ordering %d,%d,%d \n", c, a, b);
        else
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
   if (b < c)
      printf("ordering %d,%d,%d \n", a, b, c);
   else if (c < a)
         printf("ordering %d,%d,%d \n", c, a, b);
        else
         printf("ordering %d,%d,%d \n", a, c, b);
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
   if (b < c)
      printf("ordering %d,%d,%d \n", a, b, c);
   else if (c < a)
         printf("ordering %d,%d,%d \n", c, a, b);
        else
         printf("ordering %d,%d,%d \n", a, c, b);
else if  (c < b)
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
   if (b < c)
      printf("ordering %d,%d,%d \n", a, b, c);
   else if (c < a)
         printf("ordering %d,%d,%d \n", c, a, b);
        else
         printf("ordering %d,%d,%d \n", a, c, b);
else if  (c < b)
       printf("ordering %d,%d,%d \n", c, b, a);
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
   if (b < c)
      printf("ordering %d,%d,%d \n", a, b, c);
   else if (c < a)
         printf("ordering %d,%d,%d \n", c, a, b);
        else
         printf("ordering %d,%d,%d \n", a, c, b);
else if  (c < b)
       printf("ordering %d,%d,%d \n", c, b, a);
      else if (c < a)
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
   if (b < c)
      printf("ordering %d,%d,%d \n", a, b, c);
   else if (c < a)
         printf("ordering %d,%d,%d \n", c, a, b);
        else
         printf("ordering %d,%d,%d \n", a, c, b);
else if  (c < b)
       printf("ordering %d,%d,%d \n", c, b, a);
      else if (c < a)
            printf("ordering %d,%d,%d \n", b, c, a);
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
   if (b < c)
      printf("ordering %d,%d,%d \n", a, b, c);
   else if (c < a)
         printf("ordering %d,%d,%d \n", c, a, b);
        else
         printf("ordering %d,%d,%d \n", a, c, b);
else if  (c < b)
       printf("ordering %d,%d,%d \n", c, b, a);
      else if (c < a)
            printf("ordering %d,%d,%d \n", b, c, a);
           else
```

Example : ordering three numbers

```
main() {
int a,b,c;
scanf("%d,%d,%d",&a,&b,&c);
if (a < b)
   if (b < c)
      printf("ordering %d,%d,%d \n", a, b, c);
   else if (c < a)
         printf("ordering %d,%d,%d \n", c, a, b);
        else
         printf("ordering %d,%d,%d \n", a, c, b);
else if  (c < b)
       printf("ordering %d,%d,%d \n", c, b, a);
      else if (c < a)
            printf("ordering %d,%d,%d \n", b, c, a);
           else
            printf("ordering %d,%d,%d \n", b, a, c);
}
```

Switch

Switch statement matches the "expression" with one of **constant integer** values, and branches accordingly.

```
switch (expression)
{
   case const-expr: statements
   case const-expr: statements // There can be many.
   default: statements // Optional.
}
```

All **case expressions** must be different. If not, compilation error.

default case, if exists, is executed only if no case is satisfied.

break, return, goto are used to leave a switch.

Cases can be written in any order.

switch runs faster than if, especially when there are many cases.

Switch example

```
int a = 5, b = 10, sign;
if (a < b) sign= -1;
else if (a > b) sign=1;
else sign= 0;
switch (sign) {
   case -1:
   // Forget "break", and flow falls through to next case!
   // Fall-throughs must be documented and used with care.
   case 0:
     printf("Both numbers are equal to (%d)\n.", a);
     break;
   case 1:
     break;
   default:
     printf("Compare returned bad value (%d)\n.", sign);
     break;
}
```

for statement

```
for (initialization; condition; increment_decrement)
   statement // Loop body .
```

For example, to sum **factorials** from 1 to 5:

```
unsigned long int maxValue = 5;
unsigned long int sum, value, factorial = 0;
factorial=1;
// There can be several initialize statements.
for (value = 1; value <= maxValue; value++)
{
  // Watch for overflow for big numbers.
   factorial *= value;
   sum += factorial;
}

// Here sum = 153, value = 6, factorial = 120.
```

Computing x^y

```
# include <stdio.h>
main() {
unsigned long int Total-Val = 1;
```

Computing x^y

```
# include <stdio.h>
main() {
unsigned long int Total-Val = 1;
int x, y;
printf("enter two non-negative integer numbers x,y \n")
scanf( "%d,%d" ,&x,&y);
int  i;
for  (i = 1; i <= y; i + +) {
   Total-Val *= x;
}
}
```

Write a program to read a set of numbers and compute their average. Ask user to enter the number of numbers first.

Write a program to read a set of numbers and compute their average. Ask user to enter the number of numbers first.

```
main() {
float Total-Val = 0.0;

int  n,i,temp;

printf("enter number of integers \n")

scanf("%d,",&n);

for  (i = 0; i < n; i++) {

   scanf("%d,",&temp);
   Total-Val += temp;

}
printf("the average is %f \n", Total-Val/n) ;
printf("the average is %lf \n", Total-Val/n) ; // for double
}
```

What is the output of this program ?

```
# include < stdio.h >
void main() {
int i,j,n;
printf(" enter a integer number \n ");
scanf("%d", &n);
for (i = 1; i <= n; i++) {
    for (j = 0; j <= n − i; j++)
      printf(" %d ",j+1);
    printf(" \n ");
}
for (i = 1; i <= n; i++) {
    for (j = 0; j <= n − i; j++)
      printf("* ");
    printf(" \n");
}
}
```