See King Chapters 5, 6 and 7 for more details

Warning: Is your home computer a true 32-bit machine (like ohaton) or a 16-bit computer?
Do you know how to make your C/C++ compiler supports 32-bit operations?

There are several control structures in C:

Compound statement

```
{ statements }
```

Conditional statement

```
if ( expression )
    statement
if ( expression )
    statement
else
    statement
```

```
if ( expression ) {
     statements
 } else {
     statements
}
```

While statement

```
while ( expression )
    statement
```

```
while ( expression )
{ statements }
```


Do-while statement (called Do-until in Pascal)

```
do
    statement
while ( expression ) ;
```

The iteration continues when the expression evaluates to a non-zero value. A zero value ends the iterating.

For statement

```
for ( expression1 ; expression2 ; expression3 )
    statement;
```

All three expressions are optional

expression1 is evaluated once at the start of the loop
expression2 is evaluated at the start of each iteration, if the value of this expression is 0 the loop is exited
expression3 is executed at the end of each iteration.

The **for** statement is equivalent to:

```
expression1;
while ( expression2 ) {
    statement;
    expression3;
}
```


Switch statement

```
switch ( expression ) {
    case constant1 : statement
    case constant2 : statement
    case constant3 : statement
        .
        .
        .
    default : statement
} ;
```

The expression in the switch must evaluate to an integer value, all the case labels (constant) must also have integer values. When the switch statement is entered the expression is evaluated and control is transferred to the matching case (like a **goto** statement). If there is no matching case, default is used. The default case is optional. Control flows from one case to the next, *the statement is not automatically exited when the next case is reached*

Continue statement

```
continue ;
```

Causes control to transfer to the end of the smallest enclosing **while**, **do** or **for** statement. This statement is used to transfer control to the next iteration of the loop. That is, skip the current iteration and start on the next.

Break statement

```
break ;
```

The **break** statement causes the termination of the smallest enclosing **while**, **do**, **for** or **switch** statement. A **break** is commonly used to separate the cases in a switch statement.

Example: Use of the break statement

In the example we use the following strategy to read from a file or terminal

```
read first line
while ( not-at-end-of-file ) {
    process the line
    read the next line
}
```

The main problem with this approach is that we need two read statements.
An alternative schema (plan) uses the break statement

```
while ( TRUE ) {
    read the next line
    if ( at end-of-file )
       break;
    process the line
}
```