C programming language pt3

You can take as many variables as you want by using scanf:

```
printf("Enter three numbers: ");
scanf("%lf %lf %lf", &n1, &n2, &n3);
```

Decision making:

Nested if statement:

```
if (condition1)
{
    if (conditon2)
    {
        /*condition1 is true*/
        /*condition2 is true*/
    }
    else
    {
        /*condition1 is true*/
        /*condition2 is false*/
    }
}
else
{
    /*condition1 is false*/
}
```

Switch statement:

```
switch (expression)
{
case /* constant-expression */:
    /* code */
    break;

default:
    break;
}
```

It is important to put break.

You can put case and default at any order.

Looping:

For loop:

```
for (/*expression1*/;/*expression2*/;/*expresssion3*/)
{
   /* code */
}
```

Expression1: initializes counter.

Expression2: checks the condition before each iteration.

Expression3: evaluates counter after each iteration.

While loop:

```
while (condition)
{
    //code
}
```

It checks the condition before execution.

Do while loop:

```
do{
    //code
}while(condition);
```

Executes the code and then checks the condition.

Nested for loops:

```
for (short i = 0; i < count; i++)
{
    for (short j = 0; j < count; j++)
    {
        //code
    }
}
```

Break statement breaks the current loop.

Other statements:

Break statement:

```
while (condition)
{
    if (condition)
    {
        //code
        break;
    }
}
```

Used to cut out the loop.

goto statement:

```
label:
    /*block of code*/

goto label;
```

You can put label anywhere in the same function before after goto.

It is not recommended to use goto.

Continue statement:

```
for (short i = 0; i < count; i++)
{
    if (condition)
    {
        continue;
    }
    //code
}
```

Skips the current iteration.

Functions:

Used to avoid code duplication.

The functions increase modularity because functions are reusable, so it becomes maintainable.

Types of functions:

1) Library functions
   a) Open source
   b) closed source.
2) User-defined functions

Function aspects:

1. Function definition
2. Function call
3. Function declaration

Function definition:

```
return_type function_name (argument_list)
{
    //code
}
```

before or after main function

To avoid passing any argument we use void.

Every argument should have a certain type.

Function declaration:

```
return_type function_name (argument_list);
```

before main function

Function call:

```
function_name(parameter_list);
```

In main function

If function has output, it should be reserved in variable.

It is recommended that every function should have a return type. Used as error state.

Modules and multi-files approach:

In large projects, we divide program into modules.

Each module has at least two files name.c and name.h .

Header files:

1. User defined header file

```
#include <stdlib.h>
```

2. Library header file

```
#include "user.h"
```

Every function declaration is written in header files.

Every function definition is written in source files.

You include header files in source files.