#include <stdio.h> // *Tells the program you write the code in that it should read and write the C program with a specific standard (I/O library).*

int main() // *Tells the program where to start reading.*return 0; // *Returns the value 0 to the operating system and terminates the program.*

printf(”*insert string*”); // *Displays content of a string.*

scanf(”*%m.pX*”, &*variable*); // *Scans and stores a variable into the program’s memory*

if(*logical expression*) // *With if statements you can make the program read specific lines only if specific criteria have been met. By adding ”else”, you tell the program that if those specific criterias have not been met, the program will instead read a different line. You can have more than 2 statements by using ”else if”, like in the example below:  
En bild som visar text

Automatiskt genererad beskrivning*  
Logical Expressions:

== // *Equal (3=4 will be 0, 4=4 will be 1)*  
!= // *Difference (3!=4 will be 1, 4!=4 will be 0)*  
> // *Larger*  
>= // *Larger or equal*  
< // *Smaller*  
<= // *Smaller or equal*

En bild som visar text

Automatiskt genererad beskrivning

switch (*variable*) {  
case *number*: ”function”;  
 break;  
default: ”function”;  
 break;  
}

// *Similar to the if function, the switch will only execute the specific case number if the variable has the value of that number. If the variable has a value that does not correspond to any of the numbers, it will instead execute the default function. You can have several case numbers run the same function, just add them in the same line, such as ”case 2: case 21: case 42:”. The ”break” command here makes sure that the program will not run the rest of the case numbers after it matches the value with a specific case number.*

*Escape Sequences*  
\n // *Tells the program to advance to the next line.*\a // *Tells the program to play an audible ”beep” upon printing the string.*\b // *Tells the program to move the cursor back one position upon printing the string.*\t // *Tells the program to move the cursor to the next tab stop.*\” // *Tells the program to include ”” around a message, such as* printf(”\Hello!\””); *would print out the message* ”Hello!”.

int *integer*; // *Tells the program that the integer with the name ”integer” exists.*float *float*; // *Tells the program that the float with the name ”float” exists.*char *character*; // *Tells the program that the character with the name “character” exists.***Identifiers that are NOT allowed to be used as a variable:** En bild som visar text, recept

Automatiskt genererad beskrivning

variable= X; // *Gives a value of X(a number) to the variable.*variable++; // *Increases the value by 1*  
variable--; // *Decreases the value by 1*  
variable +=X; // *Increases the value by X*  
variable-=X; // *Decreases the value by X*  
variable\*=X; // *Multiplies the value by X*  
variable/=X; // *Divides the value by X*

#define TEST\_DEFINITION ”number” // *defines the constant TEST\_DEFINITION as a specific ”number”.* ***No variables can be used to define a constant.***

%m.pX (%-m.pX)

X:

%d // *Displays an integer in the decimal form (base 10). p indicates the minimum number of digits to display (extra zeros are added to the beginning of the number X if needed).*%e // *Displays a floating-point number in exponential form. p indicates how many digits should appear after the decimal point (6 by default). If p=0 then the decimal point is not displayed.*%f // *Displays a floating point in fixed decimal format, without an exponent. p has the same meaning as for %e.*  
%g // *Displays a floating point in decimal form for medium sized numbers and exponential form for very big and very small numbers. p signifies the maximum number of important digits.*%c // *Displays a single character.*

m: // *Tells how much minimum space the output should occupy, if the result is smaller than the minimum space, blank spaces will be added instead. By using -m instead of m, the blank spaces will be added from the right instead of the left.*

while(*terms*)  
{  
 ”*function*”;}  
// *The code will keep repeating the function while the terms are true. For example, while(x>4) will keep doing the function as long as the value of x is larger than 4.*

do  
{  
 ”*function*”;  
}while(*terms*);// *Similar to the while function, but it will ALWAYS do the function at least once before reading the terms. Unlike beginning with while, in the do function, the while(terms); NEEDS to end with a semi-colon (;).*

for (*variable=number;variable<number;variable++)*{

“*function*”;  
}  
// *With the for function, the program will give the variable a number, then check if the variable is smaller (or larger, or equal, any logical expression of choice) than a different number of choice and execute the function. Once the variable is no longer smaller than the number, the for function will execute “variable++” function that is written in the for(), in this case increasing the value by one.*