General form of a C program

Preprocessing directives
int main (void)
{
    /* This is a comment */
    declarations;
    statements;
}
/*Calculate the area of a triangle*/
#include <stdio.h>
int main()
{
    /*declare variables*/
    double b, h, area;
    /* enter the base and the height of the triangle*/
    printf("Enter the base and the height of the triangle: \n");
    scanf("%lf,%lf",&b,&h);
    /*calculate the area*/
    area = 0.5*b*h;
    /*output the result*/
    printf("Area = %lf",area);
    return 0;
}
C Programming

Topics Covered so far

- Data Types
- Variables
- "printf ()" function
- "scanf()" function
- Operators
C Programming

CONTROL STATEMENT
C Programming

- Used for controlling the flow of program
  - if .... else
  - for ( ; ; )
  - while
  - do while
  - switch
if Statement

- a set of statements surrounded by { and }
- The block following the if statement is executed when the condition inside () is satisfied.
- If there is no block, the next statement is executed.
- else is optional.
Print an integer number if it is between 1 & 100

```c
#include <stdio.h>
int main()
{
    int i;
    printf("Enter an integer ");
    scanf("%d",&i);

    if (i>1 && i<100)
        printf("The number is between 1 and 100.\n");
    else
        printf("The number is not in that range.\n");
    return 0;
}
```
C Programming

for Loop

- Consists of 3 parts, each separated by semicolon ( ; )
  - 1) initialization of the variable for iteration
  - 2) test of the iteration variable
  - 3) counter for iteration variable

for (initialize loop variable; condition to continue; loop variable increment)
{
    statement;
}

To write a program that computes $S=1+2+3+4+5+\ldots+100$

```c
#include <stdio.h>
int main()
{
    int i, s = 0;
    for (i=0; i<= 100; i++) s = s + i;
    printf("Sum = %d\n", s);
    return 0;
}
```
while Loop

- executes the statement(s) that follows while the condition is true
- For multiple statements, it is necessary to use a block with curly brackets `{ and }`

```c
while (condition)
{
    statement;
}
```
C Programming

Print a table showing Fahrenheit to Celsius conversion between 0°F to 100°F in steps of 10

```c
#include <stdio.h>
int main()
{
    int a;
    a = 0;
    while (a <= 100)
    {
        printf("%d degrees F = %d degrees C\n", a, (a - 32) * 5 / 9);
        a = a + 10;
    }
    return 0;
}
```
<table>
<thead>
<tr>
<th>Fahrenheit</th>
<th>Celsius</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-17</td>
</tr>
<tr>
<td>10</td>
<td>-12</td>
</tr>
<tr>
<td>20</td>
<td>-6</td>
</tr>
<tr>
<td>30</td>
<td>-1</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>60</td>
<td>15</td>
</tr>
<tr>
<td>70</td>
<td>21</td>
</tr>
<tr>
<td>80</td>
<td>26</td>
</tr>
<tr>
<td>90</td>
<td>32</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
</tr>
</tbody>
</table>
do while Loop

- very similar to the while loop except that the test occurs at the end of the loop body
- This guarantees that the loop is executed at least once before continuing.

```c
    do
    {
        statement;
    } while (condition);
```
Force a user to enter ‘0’ Or ‘1’ as input

```c
#include <stdio.h>
int main()
{
    int input_value;
    do
    {
        printf("Enter 1 for yes, 0 for no :");
        scanf("%d", &input_value);
    }
    while (input_value != 1 && input_value != 0);
    return 0;
}
```
**switch statement**

- can branch out to different tasks depending on the value of the variable used
- You can achieve the same result using multiple if statements

```c
switch(variable)
{
    case v1:
        statement 1;
        break;
    case v2:
        statement 2;
        break;
    ....
    default: (optional)
        statement 3;
        break;
}
```

Variable must be an expression of type integer or character
```c
#include <stdio.h>

int main()
{
    int a = 1;
    switch(a)
    {
    case 1:
        printf("a is 1.\n");
        printf("I am here.\n");
        break;
    case 3:
        printf("a is 3.\n");
        break;
    case 4:
        printf("a 4.\n");
        break;
    case 10:
        printf("a is 2.\n");
        break;
    }
    return 0;
}
```
Common mistakes

Wrong
if (a=0) printf(" illegal!\n");
scanf("%f", a)
for (i==0, i<10, i++)

Correct
if (a==0) printf(" illegal !\n");
scanf("%f", &a)
for (i=0; i<10; i++)