Summary of basic C-commands

compiling
To compile a c-program, you can use either cc or gcc. Whenever you use a mathematical function you have to use the -lm in addition.

cc -o executable_filename.out sourcefilename.c
gcc -o executable_filename.out sourcefilename.c
cc -o executable_filename.out sourcefilename.c -lm

For the following commands you can find at the end of this summary sample programs.
Each command in C is followed by “;”. Carriage return has no meaning in C.

comments
any text between /* and */ is ignored by the compiler. Comments are necessary for clear programs.

declaration of variables

integer variables:          int i1,i2,i3;
or for larger integers    long li1,li2;
real variables:            float f1,f2;
or for more accuracy      double d1,d2;
character variables:      char c1,c2;
pointers:                 int *pi1,*pi2;
                        double *pd1;

arrays:
int ia1[6],ia2[5][3];
double da[3];          this array consists of three double variables da[0], da[1] and da[2]
char ca[10];

assignments

integer:       i1 = 4 ;
               ia1[0] = 3 ;
               ia2[1][1] = 2 ;
real:          d1 = 4.0 ;
               da[2] = 2.0 ;
character:     c1 = "1";
pointer:       *pi1 = 3 ;
operations

arithmetic: $+$ $-$ $\ast$ $/$

comparisons:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$==$</td>
<td>equal</td>
<td>i1 == i2</td>
</tr>
<tr>
<td>$!=$</td>
<td>not equal</td>
<td>i1 != i2</td>
</tr>
<tr>
<td>$&gt;$</td>
<td>greater than</td>
<td>i1 &gt; i2</td>
</tr>
<tr>
<td>$&lt;$</td>
<td>less than</td>
<td>i1 &lt; i2</td>
</tr>
<tr>
<td>$\geq$</td>
<td>greater than or equal to</td>
<td>i1 &gt;= i2</td>
</tr>
<tr>
<td>$\leq$</td>
<td>less than or equal to</td>
<td>i1 &lt;= i2</td>
</tr>
<tr>
<td>$&amp;&amp;$</td>
<td>and</td>
<td>(i1 != i2) &amp;&amp; (i1 == i3)</td>
</tr>
<tr>
<td>$</td>
<td></td>
<td>$</td>
</tr>
</tbody>
</table>

arithmetic functions

To be able to use all of the following functions you need to compile with the option `-lm` and to write
#include <math.h>
before the function main:

dom\(x,y\) \quad x^y 
\sin(x) 
\cos(x) 
\tan(x) 
\text{asin}(x) \quad \text{sin}^{-1}(x) \text{ in range } \left[-\pi/2, \pi/2\right] 
\text{acos}(x) \quad \text{cos}^{-1}(x) \text{ in range } \left[0, \pi\right] 
\text{atan}(x) \quad \text{tan}^{-1}(x) \text{ in range } \left[-\pi/2, \pi/2\right] 
\sinh(x) 
\cosh(x) 
\tanh(x) 
\exp(x) \quad e^x 
\log(x) \quad \ln(x) 
\text{sqrt}(x) \quad \sqrt{x} 
\text{fabs}(x) \quad |x| 
\text{floor}(x) \quad \text{largest integer not greater than } x; \text{ example: floor(5.768) = 5} 
\text{ceil}(x) \quad \text{smallest integer not less than } x; \text{ example: ceil(5.768) = 6} 
\text{fmod}(x,y) \quad \text{floating-point remainder of } x/y \text{ with the same sign as } x 
x \% y \quad \text{remainder of } x/y, \text{ both } x \text{ and } y \text{ integers } (-\text{lm} \text{ is not necessary})
loops
while (x < 20) {
    sum = sum + x;
    x = x + 1;
}
for ( x=1; x < 20 ; x=x+1 )
    sum = sum + 1;

do {
    sum = sum + x;
    x = x + 1 ;}
while ( x < 20 ) ;

decision statements

if( x > 0.0 ) {
    y = 3.0/x;
    z = sqrt(x); }

if( x > 0.0 ) {
    y = 3.0/x;
    z = sqrt(x); }
else
    z = sqrt(-x) ;

if( x > 0.0 ) {
    y = 3.0/x;
    z = sqrt(x); }
else if (x < 0.0) {
    z = sqrt(-x); }
else
    z = 0 ;
output and input

To be able to use the following commands you need to write
#include <stdio.h> at the beginning of your program:

printf( "format string", variable list );

commands for format string:
\%d integer
\%f float
\%lf double
\%c character
\%s string
\n carriage return
\t tab

examples:

printf("the variable a = \%lf and i = \%d", 8.9, 3);
writes on the screen: the variable a = 8.9 and i = 3
printf("the variable i = \%d and j = \%d", 3*2, 2);
writes on the screen: the variable i = 6 and j = 2

scanf("\%lf \%lf \%d \%d", &x, &y, &i, &j);
reads in the double variables x and y and the integer variables i and j

constants

#define PI 2.0*asin(1.0) use before function main
#define N 50
functions
A function is a set of commands. All programs start execution at the function main.

At the beginning of your program you need to declare any functions:

```c
function_type function_name (types_of_parameter_list);
example: double feetinchtometer(double,double);
```

in the program you use the function with:

```c
function_name (actual_parameter_list);
exmpale: feetinchtometer(5.0,3.2);
```

After main { ... } define your function:

```c
function_type function_name (parameter_types_and_names) { declarations and statements }
exmple: double feetinchtometer(double f,double in){ ...};
```

pointers

```c
declaration: int *ip1, *ip2;
double *dp1, *dp2;
char *cp1,*cp2;
```

In the following examples assume the following declarations: int i2,i3;

assignment of pointervalues: ip1 = &i2;

assignment of values: *ip1 = 3;
*ip2 = i3;