Tiny C programs for biologists[1]

(in a not-so-random order)

Workflow :

`nano p01.c` ▶ `gcc p01.c -lm` ▶ `./a.out`

`[p01.c]` Print 'Hello world!!!' to standard output and change line (\n)

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main()
{
   printf("Hello world!!!\n");
}
```

`[p02.c]` Declare an integer variable (val), give it a value (100), and type the value of the variable to standard output :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main()
{
   int val;

   val = 100;
   printf("The value is %d\n", val );
}
```

---

1 **Why on earth there are comments containing the same exact programs ?** Copying and pasting from a PDF file does not preserve formatting and white space (like tabs and indentation), which is very bad news. The comments fix this problem : (a) Open the PDF with Adobe Acrobat Reader, you'll see the comments on the right, (b) From the three dots at the corner of a comment select "copy" to copy the program that you want, (c) in the unix shell type **`cat > myprog.c`** and hit <ENTER>, (d) Paste with right-mouse-click, (e) Hit <ENTER> and then <CTRL-D> to save the file. You now have **`myprog.c`** with correct formatting.

[p03.c] Declare a floating point variable (val), give it a value (7.15), and type its value to standard output :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main()
{
   float val;

   val = 7.15;
   printf("The value is %f\n", val );
}
```

[p04.c] Declare two variables, give them some values, and type the contents of the variables to standard output :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main()
{
   int   val1;
   float val2;

   val1 = 100;
   val2 = 7.15;
   printf("The values are %d and %f\n", val1, val2 );
}
```

[p05.c] Declare two floats, give them some values, and find and type the maximum of the two variables :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main()
{
   float val1;
   float val2;

   val1 = 3.75;
   val2 = 7.15;
   if ( val1 >= val2 )
      {
         printf("The maximum value is %f\n", val1 );
      }
   else
      {
         printf("The maximum value is %f\n", val2 );
      }
}
```

[p06.c] Instead of assigning values, attempt to read two floats from standard input using scanf(). This assumes that Babis does everything correctly :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main()
{
   float val1;
   float val2;

   scanf("%f %f", &val1, &val2 );
   if ( val1 >= val2 )
      {
         printf("The maximum value is %f\n", val1 );
      }
   else
      {
         printf("The maximum value is %f\n", val2 );
      }
}
```

[p07.c] Same as p06.c but do an extra check whether the two floats are equal (operator ==). Notice also the if() ⇒ else if() ⇒ else construct :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main()
{
   float val1;
   float val2;

   scanf("%f %f", &val1, &val2 );
   if ( val1 == val2 )
      {
         printf("Both values are equal to %f\n", val1 );
      }
   else if ( val1 > val2 )
      {
         printf("The maximum value is %f\n", val1 );
      }
   else
      {
         printf("The maximum value is %f\n", val2 );
      }
}
```

[p08.c] Read a floating point number from standard input and calculate and print its square root. No checks for negative numbers, assume Babis does everything correctly :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main()
{
   float val;
   float y;

   scanf("%f", &val );
   y = sqrt( val );
   printf("The square root of %f is %f\n", val, y );
}
```

[p09.c] The same with p08.c but check for negative numbers :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main()
{
   float val;
   float y;

   scanf("%f", &val );
   if ( val >= 0 )
      {
         y = sqrt( val );
         printf("The square root of %f is %f\n", val, y );
      }
   else
      {
         printf("Negative value detected!\n");
      }
}
```

[p10.c] Same with p09.c but confirm that Babis indeed gave us a number.

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main()
{
   float val;
   float y;

   if ( scanf("%f", &val ) == 1 )
   {
      if ( val >= 0 )
         {
            y = sqrt( val );
            printf("The square root of %f is %f\n", val, y );
         }
      else
         printf("Negative value detected!\n");
   }
}
```

[p11.c] Same with `p10.c` but with different treatment for input errors : If Babis failed to give us a number, then emit an error message and exit the program. Notice that we need an additional `#include <stdlib.h>` to use `exit()`. Notice also the operator for "not equal" (!=) :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   float val;
   float y;

   if ( scanf("%f", &val ) != 1 )
      {
         printf("Error! A number was expected. Abort.\n");
         exit( 1 );
      }

   if ( val >= 0 )
      {
         y = sqrt( val );
         printf("The square root of %f is %f\n", val, y );
      }
   else
      {
         printf("Negative value detected!\n");
      }
}
```

[p12.c] Print all integers from 0 to 99 with a step of 1. Notice the operator ++ to increase the value of a variable by 1 :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   int  i;

   for( i=0 ; i < 100 ; i++ )
      {
         printf("==> %d\n", i );
      }
}
```

[p13.c] Print all integers from 0 to 100 with a step of 2. Notice the `i += 2` expression which is used to increase the value of i by 2. Notice also the `%5d` bit in printf() which is used to type the numbers right-justified in a column with a width of 5 characters :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   int  i;

   for( i=0 ; i <= 100 ; i+=2 )
      {
         printf("==> %5d\n", i );
      }
}
```

[p14.c] For all possible combinations of integers between 0 and 10, print their product to standard output. Notice the two nested for() loops, and that that we do the multiplication inside `printf()`.

```c
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   int  i, m;

   for( i=0 ; i <= 10 ; i++ )
      {
         for ( m=0 ; m <= 10 ; m++ )
            {
               printf("%2d * %2d = %5d\n", i, m, i*m );
            }
      }
}
```

[p15.c] Print all integer numbers between 0 and 100 whose square root is less than 8.52 using a `while()` loop. Notice the use of %5.2f to print the results in the form of a table :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   float x;

   x = 0.0;
   while( sqrt(x) < 8.52 )
      {
         printf("%5.2f has a square root of %5.2f\n", x, sqrt(x) );
         x++;
      }
}
```

[p16.c] Keep reading floats from standard input, and for each float you read, type its square root in standard output. Notice the `while( scanf( …` construct. No checks for negative numbers :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   float val;

   while( scanf("%f", &val) == 1 )
      {
         printf("%8.3f has a square root of %8.3f\n", val, sqrt(val) );
      }
}
```

[p17.c] Like p16.c, but check for negative numbers :

```c
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   float val;

   while( scanf("%f", &val) == 1 )
   {
      if ( val < 0 )
         printf("%8.3f is negative. Ignored.\n", val );
      else
         printf("%8.3f has a square root of %8.3f\n", val, sqrt(val) );
   }
}
```

[p18.c] Read many floating point numbers from standard input and print the middle number from the list. For example, if Babis gave `1 3 5 10 11` we should print "`5`". Notice the use of `data[]` to store the numbers in an array (named "data") and the initialization of the value of `N`. No checks that the number of values that Babis gave us makes sense.

```c
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   float val;
   float data[ 5000 ];
   int   N, middle;

   N = 0;
   while( scanf("%f", &val) == 1 )
      {
         data[ N ] = val;
         N++;
      }

   middle = (N-1) / 2;
   printf("Read %d values. Middle entry is %f.\n", N, data[middle] );
}
```

[p19.c] Same as p18.c but with several additional checks. Notice the N%2 expression which calculates the remainder of the division of N by 2.

```c
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   float val;
   float data[ 5000 ];
   int   N, middle;

   N = 0;
   while( scanf("%f", &val) == 1 )
      {
         data[ N ] = val;
         N++;
         if ( N > 4999 )
            {
               printf("Too many values! Increase size of data[].\n");
               exit( 1 );
            }
      }

   if ( N < 1 )
      {
         printf("Less than one number given ? Abort.\n");
         exit( 1 );
      }

   if ( N % 2 == 0 )
      {
         printf("The number of values given (%d) is not odd!\n", N);
         exit( 1 );
      }

   middle = (N-1) / 2;
   printf("Read %d values. Middle entry is %f.\n", N, data[middle] );
}
```