GLG 490 Numerical Methods
Intro to C Programming – part 2

Math Library

The TA has handed out a few pages regarding the functions and constants in the math library. The handout contains the main mathematical functions and constants. When using the math library, you must remember 2 things:

1) #include <math.h>
2) You must add the –lm compile flag (i.e. gcc –lm tutorial4.c)

Try the following code named *tutorial4.c*

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

/* this program shows how to do basic math calculations */

int main(void)
{

double result;

double number1 = -36.234;
double number2 = 3.4e-4;
double number3 = 0.722;
double number4 = 7.0;

      result=fabs(number1);
      fprintf(stdout,"the absolute value of %f is %f\n",number1,result);

      result=pow(number4,number2);
      fprintf(stdout,"%f to the %f power is %f \n",number4,number2,result);


      result=asin(number3);
      fprintf(stdout,"the arcsine of %f is %f\n",number3,result);

      result=exp(number4);
      fprintf(stdout,"e to the %f is %f\n",number4,result);

      /* we can also do it this way */
      fprintf(stdout,"e to the %f is %f\n",number4,exp(number4));

      fprintf(stdout,"pi is %f\n",M_PI);

return 0;
}
```

Here is another example: *tutorial5.c*

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

int main(void)
{

/* This is a program that determines the x,y,z coordinates*/
/* along a circle of radius=1.0 along the equator (i.e. theta=pi/2). */
/* The code will print coordinates over 20 points along the circle */

double x,y,z, phi;
double delta_phi, degrees_phi;
int ipoint;

double radius=1.0;
double theta=M_PI_2;
int number_of_points=20;

/* we must determine the delta_phi for each increment */

   delta_phi=2.0*M_PI/(1.0*number_of_points);

/* loop through the increments */

   phi=0.0;
   for (ipoint=1;ipoint<=number_of_points;ipoint++)
   {

/* calculate x,y,z */
     x=radius*sin(theta)*cos(phi);
     y=radius*sin(theta)*sin(phi);
     z=radius*cos(theta);

/* convert radians to degrees for phi */

     degrees_phi=phi*180.0/M_PI;


/* the output is formatted to make the results look nicer */

  fprintf(stdout,"point: %3d  degrees: %f \n"
              "    [phi: %f theta: %f radius: %f]\n"
              "    [x: %f y: %f z: %f]\n"
              , ipoint,degrees_phi,phi,theta,radius,x,y,z);

/* We must advance to the next value of phi */
/* For this problem, theta and radius remain constant */

     phi=phi+delta_phi;
   }
return 0;
}
```

<u>More on formatting</u>: *tutorial6.c*

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

/* this program gives examples of formatting */

int main(void)
{

double number1=1.23456789e3;
int integer1=3;

/* here are some different ways to express a real number */

   fprintf(stdout,"%f %+f %15f %015f \n",number1,number1,number1,number1);

   fprintf(stdout,"\n\n");

   fprintf(stdout,"%.1f %.2f %.3f %.4f \n",number1,number1,number1,number1);

   fprintf(stdout,"\n\n");

   fprintf(stdout,"%e %E %.1e %.4e \n",number1,number1,number1,number1);

/* ok, now lets look at integers */

   fprintf(stdout,"%d %8d %.8d \n",integer1,integer1,integer1);

return 0;
}
```

<u>Opening, closing, writing, and reading files</u>

This example will show how to write to a file: *tutorial7.c*

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

/* this is a program to practice writing to files */

int main(void)
{

/* Here is the first time we see a string variable */
char temp_string[200];
char filename[200];


/* Here is the first time we see file pointer variables */
/* You must include the star immediately before the pointer name! */

FILE *fp_my_file;
```

```
int kk;

/* This is a less safe way to open a file */
/* If you couldnt open the file for some reason, the */
/* code would crash                                  */
/*
    fp_my_file=fopen("my_first_file.o","w");
*/
/* Heres a safer way */
/* If a file cannot be opened, fopen returns a NULL value*/
/* The exit command stops the program */

   if ( (fp_my_file=fopen("my_first_file.out","w")) == NULL)
   {
         fprintf(stdout,"Error - cant open file\n");
         exit(10);
   }


/* now, lets write something to it, by directing fprintf to the file pointer */

   fprintf(fp_my_file,"This is my first file\n");

/* if we are done, we must close the file.  This unconnects the pointer to the file */

   fclose(fp_my_file);

/* Now lets make a new file and write numbers to it */
/* We'll assign the name a slightly different way */

   sprintf(filename,"my_second_file.out");

/* You just assigned some text to the char variable "filename" */

   if ( (fp_my_file=fopen(filename,"w")) == NULL)
   {
         fprintf(stdout,"Error - cant open file %s\n",filename);
         exit(10);
   }

  for (kk=1;kk<=100;kk++)
  {
    fprintf(fp_my_file,"%d %f %f \n",kk,(kk*1.0),(kk*10.0));
  }

  fclose(fp_my_file);

/* Now lets say we want to add more info to "my_first_file.out" */
/* We can do this by using the "a" (append) option */
/* If you used "w" instead, you would delete all previous stuff. */

   sprintf(filename,"my_first_file.out");
   if ( (fp_my_file=fopen(filename,"a")) == NULL)
   {
         fprintf(stdout,"Error - cant open file %s\n",filename);
```

```
      exit(10);
  }

  fprintf(fp_my_file,"Here is more stuff I am adding\n");
  fclose(fp_my_file);

return 0;
}
```

Now use vi to check to make sure your files were written. Now, we will write a program to read files. Make a file named *tutorial8.c*

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

/* this is a program that reads files */
int main(void)
{

char my_file[200];
char output_file[200];
char tempstring[200];

FILE *fpin, *fpout;

int iline;
int number_to_read=50;
int ivalue;
double value;

/* Just to show that C can actually perform linux commands, lets first */
/* have the code list the contents of our directory using the linux ls command */

  system("ls");

/* Lets have C create a directory named, tutorial8_output */

  system("mkdir tutorial8_output");

/* now lets make some spaces in the output */

  fprintf(stdout,"\n\n\n\n");

/* lets create an output file, where we will write stuff to */

  sprintf(output_file,"tutorial8_output/tutorial8.out");

  if ( (fpout=fopen(output_file,"w"))==NULL)
  {
     fprintf(stdout,"cannot open %s\n",output_file);
     exit(10);
  }
/* okay, now lets open the file we created in tutorial 7 */

  sprintf(my_file,"my_second_file.out");

  if ( (fpin=fopen(my_file,"r"))==NULL)
  {
     fprintf(stdout,"cannot open %s\n",my_file);
     exit(10);
  }
  fprintf(stdout,"I just opened %s successfully\n",my_file);

/* The best way (or my way at least) to read files is to */
/* read a line of text, then parse that line into variables */

/* lets read 50 lines of the file */

  for (iline=1;iline<=number_to_read;iline++)
  {
      /* read a line of text using fgets */

      fgets(tempstring,200,fpin);

     /* So the line of information is now a string */
     /* lets test this */
      fprintf(fpout,"%s",tempstring);

     /* We use sscanf to parse that string into variables. */
     /* We choose to read the first two numbers of the line */
     /* as a integer, then a double.  We ignore the other columns */
    /* in this example      */

      sscanf(tempstring,"%d %lf",&ivalue, &value);

      /* MEMORY DANGERS - please read!!! */
      /* A couple of very important things just happened here */
      /* 1) when you want to read in a double, you MUST use %lf */
      /*    or else you will have serious memory issues        */
      /* 2) The amperstand (&) means that you are assigning the */
      /*    the value to a pointer position (see prof).  If you */
      /*    neglect the &, you will have serious memory issues  */


     /* now lets write in the values we just read to the output file*/

      fprintf(fpout,"Integer value: %d  Double value: %f\n",ivalue,value);

  }

  /* now close file */
  fclose(fpin);
  fclose(fpout);

return 0;
}
```