Computer Science 1510

Lecture 27

Lecture Outline

- File input/output

File input/output

- To access a file, we must first open the file.
- We can then read from the file, or write to the file.
- To open a file we first need a *file pointer* which contains information such as the current location within the file, the file size, etc.
- We can declare a file pointer `fp` as follows:

```
FILE *fp;
```

- We use this pointer to open a file using the `fopen` function, which is declared in `stdio.h`.
  Syntax:

```
fp=fopen(filename,access)
```

  where `filename` is the name of the file to open (in double quotes), and `access` is a string (in double quotes) indicating how the file will be used.

File input/output

- For example, to open a file called `names.dat` for reading, we would have:

```
fp=fopen("names.dat","r");
```

- The following is a list of access codes that can be used when opening a file:

`r`	Open file for reading, error if file doesn't exist.
`w`	Open file for writing, over-write if file exists.
`a`	Append to the file, create file if it doesn't exist.
`r+`	Open file for reading and/or writing.
`w+`	Create file for reading and/or writing.
`a+`	Open or create file for appending.

- The `fopen` function returns NULL if it was not successful.

- To close a file in C, we use the `fclose` command as follows:

```
fclose(fp);
```

File input

- To read from a file we can use the `fscanf` command, declared in `stdio.h`.

- Syntax:

  ```
  fscanf(file-pointer,description,&variable-list)
  ```

  where `file-pointer` corresponds to the file to be read, `description` contains conversion codes describing what will be read, and `&variable-list` contains the locations where the read values will be written to memory.

- The fscanf function returns an integer equal to the number of items successfully converted, or a value of `EOF` if the end of the file has been reached.

Example: File input

```
#include <stdio.h>

int main(int argc, char *argv[])
{
   int i,numtimes=50;
   double failure_time[50],sum,mean;
   char filename[20];
   FILE *fp=NULL;

   /* Read the filename containing the data */
   printf("Which file would you like to read failure times from?\n");
   scanf("%s",filename);

   /* Open file from which to read the failure times */
   fp=fopen(filename,"r");
   if (fp==NULL) {
       printf("Unable to open file %s\n",filename);
       return -1;
   }

   /* Read the failure times and store them in array failure_time */
   for (i=0;i<numtimes;i++){
      fscanf(fp,"%lf",&failure_time[i]);
   }
   fclose(fp); /* Close the file */

   /* Calculate the mean time to failure */
   sum=0.0;
   for (i=0;i<numtimes;i++){
      sum+=failure_time[i];
   }
   mean = sum/numtimes;
   printf("Mean time to failure = %lf\n",mean);

   /* Print list of failure times greater than the mean */
   printf("List of failure times greater than the mean:\n");
```

```
   for (i=0;i<numtimes;i++){
      if (failure_time[i] > mean) printf("%lf\n",failure_time[i]);
   }
   return 0;
}
```

Example: Histogram

```
#include <stdio.h>

int main(int argc, char *argv[]){
    float failure;
    int i, j, num_bins=6, num_pts, bins[num_bins], ret=0;
    FILE *fp=NULL;

    /* Check that the user entered the filename */
    if (argc!=2){
        printf("USAGE: %s filename\n",argv[0]);
        return -1;
    }

    /* Open the specified file */
    fp = fopen(argv[1],"r");
    if (fp==NULL){
        printf("Unable to open file %s\n",argv[1]);
        return -2;
    }

    /* Initialize bins */
    for (i=0;i<num_bins;i++){
        bins[i]=0;
    }

    num_pts = 0;
    /* Read failure time and increment appropriate bin */
    ret = fscanf(fp,"%f",&failure);
    while (ret != EOF) {   /* Read to end of file */
        num_pts++;
        if (failure < 1.0) {
            bins[0]++;
        } else if (failure >= 1.0 &&  failure < 2.0) {
            bins[1]++;
        } else if (failure >= 2.0 &&  failure < 3.0) {
            bins[2]++;
```

```
        } else if (failure >= 3.0 &&  failure < 4.0) {
            bins[3]++;
        } else if (failure >= 4.0 &&  failure < 5.0) {
            bins[4]++;
        } else if (failure >= 5.0) {
            bins[5]++;
        }
        ret = fscanf(fp,"%f",&failure);
    }

    /* Draw histogram */
    for (i=0;i<num_bins;i++) {
        if (i != num_bins-1) {
            printf("[%d, %d): ",i,i+1);
        } else {
            printf("[%d, infinity): ",i);
        }
        for (j=0;j<bins[i];j++){
            printf("*");
        }
        printf("\n");
    }

    fclose(fp);
    return 0;
}
```

Example: Histogram 2

```
#include <stdio.h>

int main(int argc, char *argv[]){
    double failure, bin_size;
    int i, j, num_bins=6, num_pts, bins[num_bins], ret=0;
    FILE *fp;

    /* Check that the user entered the filename */
    if (argc!=2){
        printf("USAGE: %s filename\n",argv[0]);
        return -1;
    }

    /* Open the specified file */
    fp = fopen(argv[1],"r");
    if (fp==NULL){
        printf("Unable to open file %s\n",argv[1]);
        return -2;
    }

    /* Initialize bins */
    for (i=0;i<num_bins;i++){
        bins[i]=0;
    }

    /* Determine size of each bin */
    printf("What should the bin size be?\n");
    scanf("%lf",&bin_size);

    num_pts = 0;
    /* Read failure time and increment appropriate bin */
    ret = fscanf(fp,"%lf",&failure);
    while (ret != EOF) {   /* Read to end of file */
        num_pts++;

        /* Increment appropriate bin */
```

```
        for (i=num_bins-1;i>=0;i--) {
            if (failure >= i*bin_size) {
                bins[i]++;
                break;
            }
        }
        ret = fscanf(fp,"%lf",&failure);
    }

    /* Draw histogram */
    for (i=0;i<num_bins;i++) {
        if (i != num_bins-1) {
            printf("[%lf, %lf): ",i*bin_size,(i+1)*bin_size);
        } else {
            printf("[%lf, infinity): ",i*bin_size);
        }
        for (j=0;j<bins[i];j++){
            printf("*");
        }
        printf("\n");
    }

    fclose(fp);
    return 0;
}
```

File output

- To write to a file we can use the `fprintf` command, also declared in `stdio.h`.

- Syntax:

```
fprintf(file-pointer,description,variable-list)
```

  where `file-pointer`, `description` and `variable-list` are similar to those for `fscanf`.

Example: File output

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

int main(int argc, char *argv[]){
    float x, minvalue, maxvalue, inc;
    int nvals;
    FILE *fp=NULL;

    if (argc!=2) {
        printf("USAGE: %s filename\n",argv[0]);
        return -1;
    }
    fp = fopen(argv[1],"w");
    if (fp == NULL) {
        printf("Unable to open file %s\n",argv[1]);
        return -2;
    }

    printf("What is the starting value?\n");
    scanf("%f",&minvalue);
    printf("What is the ending value?\n");
    scanf("%f",&maxvalue);
    printf("How many values would you like in the table?\n");
    scanf("%d",&nvals);
    fprintf(fp, "x       sin(x)    cos(x)\n");
    fprintf(fp, "-------------------------\n");

    inc = (maxvalue-minvalue)/(nvals-1); /* Compute the increment */
    x = minvalue;
    do {
        fprintf(fp, "%6.2f %10.5f %10.5f\n",x,sin(x),cos(x));
        x = x + inc;
    } while (x<=maxvalue);

    fclose(fp);
    return 0;
}
```