CS 101
Computer Programming and utilization

IIT BOMBAY

Dr Deepak B Phatak
Subrao Nilekani Chair Professor
Department of CSE, Kanwal Rekhi Building
IIT Bombay

Session 13, Strings and Files

Friday, September 9, 2011

Overview

- More on parameter passing
- Formatted Output and input
  - Functions printf() and scanf()
- More string functions
  - strcpy, strcmp, sprintf, sscanf
- File handling in C++
  - Files on disk
  - Sequential text files
  - Binary files
  - Direct access file
- Announcements

Passing parameters to functions

```
int  f (int x, int y){
        int sum;
        sum = 5*x + y%10;
        return sum;
    }
```

To use this function in our program

```
int A, B, C;
cin >> A  >> B;
C = f(A, B);
cout << C;
```

Parameter passing

- Normally, parameters are passed by ‘value’
  - Values of actual parameters are copied to locations of formal parameters,
    - when the function is invoked
  - Only the return value comes back.
- Significant overhead of copying
  - What if a parameter is a large array?

Parameter passing …

- We may wish to change some parameter values inside the function, and send these back to the actual parameters in the calling program
  - Example: Solving simultaneous equations

```
int  simsolve (int N, float C[100,100], float B[100], float X[100]);
```

- We will supply the coefficient matrix C, and the RHS array B
  - We wish to get back the result array X. How?
- The type declared for the function (int), which describes the type of value normally returned, has no relevance

Function to swap values in locations a and b

```
int swap (int a, int b){
    int temp;
    temp = a; a= b; b = temp;
    return 0;
}
```

- In the main program, we may write:

```
int dummy, X, Y; X= 5; Y =10;
dummy = swap(X,Y);
```

Passing by reference

- Alternate mechanism for passing parameters
  - A pointer can be passed to the function
  - Operations within the function will be carried out directly on the actual parameters
  - No copying,
  - Parameter logically ‘returns’ back

Passing by reference …

```
int main() {
   int x=5, y=10;
   swap(&x, &y);
   cout << "x=" << x
   cout << " y=" << y << “\n”;
 }
```

Passing by reference

```
void swap(int *a, int *b) {     // call by reference
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
  }
```

Passing Array parameters

- Arrays, passed to a function, come back with modified values.
  - Why?
- Array name, when passed as a parameter, is automatically treated as a pointer to the 0th element of the array.

I/O functions in C++

- C++ does not have any instruction for reading or writing data
- How is input/output to external world handled?
  - Using cin and cout

```
cin   >> x >> y;
z = 2.5 * x -7.32 * y;
cout  << “Value of z is: ”  << z ;
```

printf(), scanf()

- Special functions in C++ library to perform formatted input and output

  scanf() and printf()

- Parameters to these functions include a “format” string, followed by data values (expressions) to be read/printed
- C++ applies the appropriate pattern to each value
  - for interpreting characters in the input string and converting these to values
  - for generating output string from given values

printf(“format-string”, value, value, …)

- This function displays one or more values on the user terminal

  printf("%d is a number\n", N);
- If value of N is, say, 523, output produced by this function call is:

  523 is a number

- The format string has a “format specifier” (%d), which is used to interpret N and convert it to a formatted value. Other characters are displayed as they are. \n introduces a new line
- Specifiers can appear anywhere, each must correspond to a value appearing after the format string

Examples of format specification

%6d - 6 digit integer

%7s - string fitted in 7 characters space

%8.2f - float, 8 digits total, 2 after decimal point

%8.2g - same, switch to E notation if required

scanf(“format-string”, &var, &var, ..)

- scanf needs to be passed *pointers* to its arguments, so that the values read can be assigned to the proper destinations.
- Forgetting to pass a pointer is a very common error, and one which the compiler cannot detect
- The “format-string” is used to control interpretation of input characters (bytes containing ASCII codes)
- These input bytes contain values to be assigned to the objects pointed to by the remaining arguments to scanf.

Examples of scanf

```
int main(){
    int M, N; float x, y; char name[40];
    scanf("%d %d",&M, &N);
    cout<<M<<" "<<N<<endl;
    scanf(:"%f %f %s", &x, &y, name);
    cout<< x << " " << y <<endl;
    cout << name <<endl;
    return 0;
}
```

Another example of scanf

```
int main(){
    int a; float x; char itemcode[8];
    // The input data line contains
    //  123456fanbelt150.50
    scanf("%6d%7s%f", &a, itemcode, &x);
    printf("%6d %7s %f\n", a, itemcode, x);
    return 0;
}
```

Other string functions

- strcmp,
  - Compare two strings
- strcpy
  - Copying one string into another
- sprintf
  - ‘print’ formatted values onto a string, rather than to terminal

Using sprintf()

```
char   s[75];
int roll = 12345,  int batch =112;
….
sprintf( s, “%5d %3d\n”, roll, batch):
…
```

This will produce the following string in s

“12345 112\n”

Files

- A File is regarded as a large collection of bytes, stored outside the main memory, typically on a disk.
- Files on the disk are managed by the OS
  - By a component called file system.
- A new file can be created, data can be written to it, and data can be retrieved from it
- Data can be inserted into or deleted from an existing file

File properties

- Each file has certain properties.
  - It has a name (and extension), a ‘path’, permissions, etc.
- Physical location of a file and its properties, are known to the OS
- Logically, C++ treats a file as a large array of bytes
  - May contain character data (Text files)
  - May contain ‘encoded’ data (binary files)

Handling file within a program

- A file is handled through a file pointer, declared by

  FILE *fp;

  FILE *infile, *outfile;

- A file has to be opened. Records can then be read or written to the file. It should be closed at the end

Handling text files

- Consider a file

10101,Anil,112,
10102,Amit,111,
10103,Shefali,112,
10104,Rajesh,111,
10105,Nandan,111,
10106,Avinash,112,
10107,Srikant,112,
10108,Nilmani,111,
10110,Rajesh,112,
10115,Ketan,111,

Problem

- Read records (lines) from the text file, extract the three strings from each record in three separate arrays.
- Then construct a single string containing these three parts
  - Ensure that these parts have exactly 5, 29, and 3 characters

Algorithm Logic for our program

A sample line in the file is shown below:

10108,Nilmani,111,

read_text_file.cpp

```
int main() {
  char linestr[80];
  char sroll[6], sname[30], sbatch[4];
  int i,j,k,N =0;
  FILE *fp;
  fp = fopen("inputdata.txt", "r" );
  if (fp == NULL){
     cout << "Could not open file" << endl;
     return -1;
  }
  /* file is opened at this point. Each record will be read,
     then split into three parts, and these parts will be printed*/
```

Program …

```
fgets(linestr, 79, fp);
while (!feof (fp)){
    /* valid string, separate the parts */
    i =0; k =0;
    while ((sroll[i++] = linestr[k++]) != ',');
    sroll[i-1]='\0'; i=0;
    while ((sname[i++] = linestr[k++])!= ',');
    for (j = i-1; j<29; j++) sname[j] = ' ';
    sname[29] ='\0'; i=0;
    while ((sbatch[i++] = linestr[k++]) != ',');
    sbatch[i-1] = '\0';
```

Program …

```
        cout << N++ << "  " << sroll << "  ";
        cout << sname << "  " << sbatch << endl;
        fgets(linestr, 79, fp);
    }
    cout << "inputfile has been read and printed\n";
    cout << endl << endl;
    fclose(fp);

}
```

Program “read_write_text_files.cpp”

```
#include <iostream>
#include <cstring>
using namespace std;
int main() {
  char linestr[80]; char outstr[80];
  char sroll[6], sname[30], sbatch[4];
  int i,j,k,N =0;
  FILE *fp; FILE* fpout;
```

Program …

```
 fp = fopen("inputdata.txt", "r" );
 if (fp == NULL){
   cout << "Could not open file" << endl;
   return -1;
}

fpout = fopen ("studentdb.txt", "w");
if (fpout == NULL){
   cout << "Could not create output file" << endl;
   return -1;
}
```

Program …

```
/*Input  file is open at this point, read records one by one */
  fgets(linestr, 79, fp);
  while (!feof (fp)){
    /* valid string, separate the parts */
    i =0; k =0;
    while ((sroll[i++] = linestr[k++]) != ',');
    sroll[i-1]='\0'; i=0;
    while ((sname[i++] = linestr[k++])!= ',');
    for (j = i-1; j<29; j++) sname[j] = ' ';
    sname[29] ='\0'; i=0;
    while ((sbatch[i++] = linestr[k++]) != ',');
    sbatch[i-1] = '\0';
```

Program

```
    /* prepare output string and write to database*/
    sprintf(outstr, "%2d %5s %30s %3s\n",N,sroll,sname, sbatch);
    fputs(outstr,fpout);
    fgets(linestr, 79, fp);
    N=N+1;
  }
  cout << "inputfile has been read and printed\n";
  cout << " studentdb.txt file created\n";
  fclose(fp);  fclose(fpout);
  return 0;
}
```

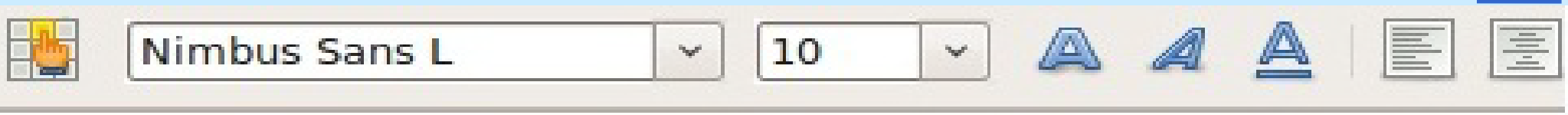
Execution results

```
$a.out
inputfile has been read and printed
 studentdb.txt file created
$
```

Results …

```
$cat studentdb.txt
 0 10101  Anil                112
 1 10102  Amit                111
 2 10103  Shefali             112
 3 10104  Rajesh              111
 4 10105  Nandan              111
 5 10106  Avinash             112
 6 10107  Srikant             112
 7 10108  Nilmani             111
 8 10110  Rajesh              112
 9 10115  Ketan               111
$
```

A sample spread sheet



	A	B	C	D	E
1	10101	Anil Shah	112	12.5	
2	10102	Amit Jadhav	111	15	
3	10103	Shephali Pandya	112	17	
4	10104	Rajesh Mashruwala	111	19	
5	10105	Nandan Meshram	111	16	
6	10106	Avinash Adsule	112	14	
7	10107	Srikant Rao	112	14.5	
8	10108	nilmani Raut	111	11.5	
9	10110	Rajesh Singh	112	10	
10	10115	Ketan Maheshwari	111	12	
11					
12					

Comma Separated Values format (.csv)

IIT BOMBAY

10101,Anil Shah,112,12.5

10102,Amit Jadhav,111,15

10103,Shefali Pandya,112,17

10104,Rajesh Mashruwala,111,19

10105,Nandan Meshram,111,16

10106,Avinash Adsule,112,14

10107,Srikant Rao,112,14.5

10108,Nilmani Raut,111,11.5

10110,Rajesh Singh,112,10

10115,Ketan Maheshwari,111,12

Announcements

- Midsem Exam

- Schedule:

  Date and Time: Friday, 16th September 2011, 08:30 to 10:30

  Venue: Convocation Hall.

- You will have to check the exact seating arrangement which will be displayed in the Convocation Hall foyer, and take your seat
- Please assemble in the hall foyer latest by 08:15
- Midsem Exam will be open notes like the quiz.

- Syllabus: Every thing covered up to this lecture, except files

Programming Contest

- Programming Contest for CS101 1st Semester

(Curtsey: Mr Avinash Awate, Program director, Ekalavya project)

- Produce an industry standard program to output IIT JEE ranking
  Inputs :

  I.1) A file containing Roll number (7 digits) and marks in PCM
  I.2) Cutoff percentage (Qualifying top %ile in each rank)
  I.3) Number of merit list ranks to be output

Outputs:

O.1) Merit list comprising Rank and Roll number
O.2) Total number of candidates in merit list
O.3) Errors (invalid input and abnormal conditions)

Contest …

- You have to submit a file containing two functions

```
int readinput(FILE *inputfp)
        // returns number of valid student records
int meritlist(int cutoffpercentile, int ranksrequired)
        // returns number of students in merit list
```

Your code will call the following functions:

```
void printerror(int errornum, int linenum)
        // prints errors during processing
void printmerit(int rank, long rollnum)
        // prints a merit rank
```

Contest …

- Prizes to top 3 programmers of each slot (two sets)

  ` 5000, 3000, 2000

  If we get excellent quality programs, a further (timed) contest will be held for the 6 winners to decide the CODEMATSER of semester I

  DETAILED SPECIFICATION (Error Numbers/Test Files/Sample code) AVAILABLE on Moodle
  Queries to be sent to Anup Naik (anupnaik[at]cse.iitb.ac.in)

  Submission deadline: 2nd January 2012