More on Arrays

Abhiram Ranade

Topics

- Text processing: C style
  - C++ style: later.
- Two dimensional arrays

Text processing

Simple example:

Marks display program: use names of students instead of roll numbers.

More complex example:

Read in an entire book. Allow users to search for text. Exact match? Approximate match?

Read in a C++ program. Compile it.

How to store text on a computer

- Use an array of char

```
char nameOfBook[50], nameOfAuthor[50];
```

- What size array to allocate?

  People do not have names of same length

  “Om Puri”, “Chakravarti Rajagopalachari”

- C and C++ handle this problem differently.

Simple solution (“C style”)

- Allocate an array of largest possible size
- Use only as much from it as needed.

Need to remember what portion of the array is being used.

- If text stored has L characters, then store text in indices 0..L-1. Store ASCII value 0, i.e. ‘\0’ at index L.
- ASCII value 0 is reserved for this purpose, cannot be used in normal text.

Storage Scheme Example

char name[50];

- Storing “Om Puri”:

name[0]: ‘O’, name[1]: ‘m’, name[2]: ‘ ‘

name[3]: ‘P’, name[4]: ‘u’, name[5]: ‘r’,

name[6]: ‘i’, name[7]: ‘\0’

name[8..49]: not used.

- Storing “Chakravarti Rajgopalachari”

name[0]: ‘C’ … name[25]: ‘i’, name[26]: ‘\0’

name[27..49]: unused.

How to actually store: 1

- Initialize while defining the array:

```
char name[50] = “Om Puri”;
```

This will store all 7 letters and also ‘\0’.

How to actually store: 2

- Read in from keyboard

```
char name[50];          // in general char name[n];
cin.getline(name,50); // in general ..(name,n)
```

Will read in what you type and store into consecutive elements of name, until either

- 49 characters have been read (i.e. n-1)
- end-of-line/enter has been read.

Then '\0' will be stored.

An unsafe mechanism

- Read in from keyboard

```
char name[50];
cin >> name;
```

Will store the first “word” you type into name, followed by ‘\0’. word: white space delimited.

- “Om Puri” : only “Om” will be stored.
- If “word’ has length 50 or more, index out of bounds error.

Printing out strings

```
char name[50];
....
cout << name;
```

Will print out characters in name till '\0'.

```
cout << "The name is: ";
```

"…" : character string constant, can be used wherever char array name is expected.

Modifying text strings

```
char subject[20] = “Biology”;
subject[0] = ‘Z’;  subject[1] = ‘o’;
cout << subject;
```

- What will this print?

- Cannot “assign” multiple characters in one statement.

Some basic operations on strings

- Finding the length of a string stored in a char array. String length might be < array length.
- Copying a string stored in one array into another array. Need not copy entire array.
- Comparing two strings for equality.
- Deciding which string appears first in the dictionary. “lexicographic order”.

Text processing: general idea

Like processing ordinary arrays. “Process elements starting at 0 till ‘\0’ is found”

Length of a string

```
char name[50];  int L;
cin.getline(name,50);
for(L=0; name[L] != '\0'; L++){}
cout << "name has length "<< L << endl;
```

Will L always be in range?

Can you write this as a function?

Copying a string

```
//Preconditions: source must contain ‘\0’
// terminated string, dest must be long enough.
void strcpy(char dest[], char source[]){
  int i;
  for(i=0; source[i] != '\0'; i++)
        dest[i]=source[i];
  dest[i]=source[i];    // copy the '\0' itself
}  // Write a main program to use this.
```

Lexicographic comparison

```
char compare(char a[], char b[]){
  for(int i=0;  ; i++){
    if(a[i] == '\0' && b[i] == '\0') return '=';
    if(a[i] == '\0') return '<';
    if(b[i] == '\0') return '>';
    if(a[i]<b[i]) return '<';
    if(a[i]>b[i]) return '>';
}}  // lexicographic = Dictionary order
```

Two dimensional arrays

- Arrays: useful for storing sequences. "mathematical object having one subscript"
- 2 dimensional arrays: for storing matrices. "mathematical object having two subscripts".

```
double a[10][20];
```

200 elements: a[0][0], a[0][1], ..., a[0][19], a[1][0], a[1][1], … a[9][18], a[9][19]. Stored in this order in memory.

Defining with initialization

```
double a[3][2]={{1,2},{3,4},{5,6}};

for(int i=0; i<3; i++){
   for(int j=0; j<2; j++) cout << a[i][j] << " ";
   cout << endl;
}
```

Example: Matrix multiplication

```
double a[3][2]={{1,2},{3,4},{5,6}}, b[2][4]=
   {{1,2,3,4},{5,6,7,8}},   c[3][4];


for(int i=0; i<3; i++)
   for(int j=0; j<4; j++){
      c[i][j] = 0;
      for(int k=0; k<2; k++) c[i][j] += a[i][k]*b[k][j];
   }
```

Using only one index

```
void printrow(double p[], int n){
  for(int i=0; i<n; i++) cout << p[i] << endl;
}
int main(){
double a[10][20];  … ; printrow(a[4],20);}
```

a[i] : one dimensional array corresponding to the i th row of the matrix.

Example

```
char countries[6][20] = {"India", "China",
  "Sri Lanka", "Nepal”,  "Bangladesh", "Pakistan”}
cout << compare(countries[0], countries[1]);
```

will print ‘>’ because India is after China in the dictionary order.

Two dimensional arrays and functions

```
void printCountries(char countries[][20], int n){
  for(int i=0; i<n; i++)
     cout << countries[i] << endl;
}
int main(){
  char countries[6][20] = {…}
  printCountries(countries, 6);
}
```

Remarks

- Second dimension of an array that is a parameter to a function must be a compile time constant.
- Compile-time constant: an expression whose value is known at compile time. Expressions are allowed, e.g. 5*n+6 where n is defined as

```
const int n=6;
```

Drawing Polygons

Polygon pname(double cx, double cy,
    double vertices[][2], int n);

Draws an n vertex polygon named pname

cx, cy: where local origin of polygon is placed.

vertices: Each row gives coordinates of a vertex relative to local origin.

n: number of vertices, first dimension of array vertices.

Drawing Polygons

```
int main(){
   initCanvas("Pentagon");
   double penta[5][2] = {{-50,50}, {50,50},
            {100,0}, {0,-50}, {-100,0}};
   Polygon p(250,250, penta, 5);
   for(int i=0; i<36; i++) p.left(10);
   getClick();
}
```

Remarks

- Edges of the polygon may intersect themselves. “correct thing is done”. Try it out.
- forward, left, scale, setColor, setFill etc. also work.

Exercise

Read in strings into a 2d array.

Print them out in lexicographic order.

- Note that once strings are in lexicographic order, we can do binary search over them!