

CS111, C Programming

Lab / Array

黄嘉炜

huangjw3@mail.sustech.edu.cn



深港微电子学院
SCHOOL OF MICROELECTRONICS



Outline

- Review
- Array: Showcase, String
- Array: Showcase, 2D
- Assignment



Review: Date Differential

Bug ?

```
21  int dateDiff(int year1, int month1, int day1, int year2, int month2, int day2) {  
22  
23      int days1 = dayOfYear(year1, month1, day1);  
24      int days2 = dayOfYear(year2, month2, day2);  
25  
26      for (int y = year1; y < year2; y++) {  
27          days2 += isLeapYear(y) ? 366 : 365;  
28      }  
29  
30      return days1 > days2 ? days1 - days2 : days2 - days1;  
31  }  
--
```



Review: Date Differential

Bug ?

```
5  int y1, m1, d1, y2, m2, d2, tempy, tempm, tempd, diffy, diffm, diffd;  
6  scanf("%d-%d-%d", &y1, &m1, &d1);  
7  scanf("%d-%d-%d", &y2, &m2, &d2);  
8  if (y2 > y1 || (y1 == y2 & m2 > m1) || (y1 == y2 & m1 == m2 & d2 > d1))//调换顺序, 使y1大于y2,  
9  {  
10     tempy = y1;  
11     tempm = m1;  
12     tempd = d1;  
13     y1 = y2;  
14     d1 = d2;  
15     m1 = m2;  
16     y2 = tempy;  
17     m2 = tempm;  
18     d2 = tempd;  
19 }
```

Review: Input non-string array

Better way ?

```
3  int arr[10];  
4  scanf("%d%d%d%d%d%d%d%d%d%d",  
5      &arr[0], &arr[1], &arr[2], &arr[3], &arr[4],  
6      &arr[5], &arr[6], &arr[7], &arr[8], &arr[9]);
```



Review: Input non-string array

Better way !

```
4   int n[10];  
5   int max , min , repeatTime = 0;  
6  
7   for (int i = 0; i < 10; i++)  
8   {  
9       scanf("%d", &n[i]);  
10  }
```

```
4   int inputCard[17];  
5   int cardDictionary[15] = {0};  
6   for (int i = 0; i < 17; i++)  
7   {  
8       scanf("%d", &inputCard[i]);  
9   }
```

Outline

- Review
- **Array: Showcase, String**
- Array: Showcase, 2D
- Assignment



Array: Showcase, String

String declaration & initialization

```
8   char str1[] = "Hello World!";
9   printf("sizeof(str1): %d\n", sizeof(str1));
10  for (int i = 0; i < sizeof(str1); i++) {
11      printf("str1[%d] = %c (%d)\n", i, str1[i], str1[i]);
12  }
```

```
sizeof(str1): 13
str1[0] = H (72)
str1[1] = e (101)
str1[2] = l (108)
str1[3] = l (108)
str1[4] = o (111)
str1[5] =   (32)
str1[6] = W (87)
str1[7] = o (111)
str1[8] = r (114)
str1[9] = l (108)
str1[10] = d (100)
str1[11] = ! (33)
str1[12] = ??
```


Array: Showcase, String

Length of String, with help of: `strlen(...)` in `<string.h>`

```
1  #include <stdio.h>
2  #include <string.h>
```

```
8      char str1[] = "Hello World!";
```

```
14     printf("strlen(str1): %d\n", strlen(str1));
15
```

strlen, strlen_s

Defined in header `<string.h>`

```
size_t strlen( const char *str );
```

(1)

- 1) Returns the length of the given null-terminated byte string, that is, the number of characters in a character array whose first element is pointed to by `str` up to and not including the first null character.

The behavior is undefined if `str` is not a pointer to a null-terminated byte string.

Array: Showcase, String

Length of String, with help of: `strlen(...)` in `<string.h>`

```
1  #include <stdio.h>
2  #include <string.h>
```

```
8      char str1[] = "Hello World!";
```

```
14     printf("strlen(str1): %d\n", strlen(str1));
15
16     str1[5] = '\0';
17     printf("sizeof(str1): %d\n", sizeof(str1));
18     printf("strlen(str1): %d\n", strlen(str1));
19     printf("%s\n", str1);
```

```
strlen(str1): 12
sizeof(str1): 13
strlen(str1): ??
```

??

Array: Showcase, String

Input a string from stdin

```
21 char str2[100] = {0};
22 printf("after define, sizeof(str2): %d\n", sizeof(str2));
23 printf("after define, strlen(str2): %d\n", strlen(str2));
24
25 // scanf("%s", &str2);
26 scanf("%s", str2); // bug ?
27 printf("after input, sizeof(str2): %d\n", sizeof(str2));
28 printf("after input, strlen(str2): %d\n", strlen(str2));
29 printf("%s\n", str2);
```

```
after define, sizeof(str2): 100
after define, strlen(str2): 0
hello
after input, sizeof(str2): 100
after input, strlen(str2): 5
hello
```



| Array: Showcase, String

Input a string from stdin

```
printf("value of str2: %x\n", str2);  
printf("pointer of str2: %p\n", &str2);
```

```
value of str2: 61fd90  
pointer of str2: 000000000000061FD90
```



Array: Showcase, String

Input a string from stdin: **DO NOT use scanf !**

```
21 char str2[100] = {0};
22 printf("after define, sizeof(str2): %d\n", sizeof(str2));
23 printf("after define, strlen(str2): %d\n", strlen(str2));
24
25 // scanf("%s", &str2);
26 scanf("%s", str2); // bug ?
27 printf("after input, sizeof(str2): %d\n", sizeof(str2));
28 printf("after input, strlen(str2): %d\n", strlen(str2));
29 printf("%s\n", str2);
```

```
after define, sizeof(str2): 100
after define, strlen(str2): 0
hello world!
after input, sizeof(str2): 100
after input, strlen(str2): 5
hello
```



Array: Showcase, String

Input a string from stdin: plz use fgets / gets !

fgets

Defined in header <stdio.h>

```
char *fgets( char          *str, int count, FILE          *stream );    (until C99)
char *fgets( char *restrict str, int count, FILE *restrict stream );    (since C99)
```

Reads at most `count - 1` characters from the given file stream and stores them in the character array pointed to by `str`. Parsing stops if a newline character is found, in which case `str` will contain that newline character, or if end-of-file occurs. If bytes are read and no errors occur, writes a null character at the position immediately after the last character written to `str`.

Parameters

str - pointer to an element of a char array
count - maximum number of characters to write (typically the length of `str`)
stream - file stream to read the data from

Return value

`str` on success, null pointer on failure.

Array: Showcase, String

Input a string from stdin: plz use fgets / gets !

```
21 char str2[100] = {0};
22 printf("after define, sizeof(str2): %d\n", sizeof(str2));
23 printf("after define, strlen(str2): %d\n", strlen(str2));
24
25 // scanf("%s", &str2);
26 // scanf("%s", str2); // will stop input when space/enter
27 fgets(str2, 100, stdin);
28 printf("after input, sizeof(str2): %d\n", sizeof(str2));
29 printf("after input, strlen(str2): %d\n", strlen(str2));
30 printf("%s\n", str2);
```

```
after define, sizeof(str2): 100
after define, strlen(str2): 0
hello world!
after input, sizeof(str2): 100
after input, strlen(str2): 13
hello world!
```

Note: will including \n in input string!

Array: Showcase, String

Input a string from stdin: plz use fgets / gets !

gets, gets_s

Defined in header <stdio.h>

```
char *gets( char *str );
```

 (1) (removed in C11)

```
char *gets_s( char *str, rsize_t n );
```

 (2) (since C11)

1) Reads `stdin` into the character array pointed to by `str` until a newline character is found or end-of-file occurs. A null character is written immediately after the last character read into the array. The newline character is discarded but not stored in the buffer.

Parameters

str - character string to be written

Return value

`str` on success, a null pointer on failure.

If the failure has been caused by end of file condition, additionally sets the `eof` indicator (see `feof()`) on `stdin`. If the failure has been caused by some other error, sets the `error` indicator (see `ferror()`) on `stdin`.

Array: Showcase, String

Input a string from stdin: plz use fgets / gets !

```
27 // fgets(str2, 100, stdin); // will contains enter (\n)
28 gets(str2);
29 printf("after input, sizeof(str2): %d\n", sizeof(str2));
30 printf("after input, strlen(str2): %d\n", strlen(str2));
31 printf("%s\n", str2);
```

```
hello world!
after input, sizeof(str2): 100
after input, strlen(str2): 12
hello world!
```

Array: Showcase,

Toggle case (大小写互换)

```
Hello World!  
hELLO WORLD!  
count of alphabet: 10
```

```
8      char str[MAX_LEN];  
9      fgets(str, MAX_LEN, stdin);  
10  
11     int len = strlen(str);  
12     int alphabetCnt = 0;  
13     for (int i = 0; i < len; i++) {  
14         // When A ~ Z  
15         if (str[i] >= 'A' && str[i] <= 'Z') {  
16             str[i] = ??  
17             alphabetCnt += 1;  
18         }  
19         // when a ~ z  
20         else if (str[i] >= 'a' && str[i] <= 'z') {  
21             str[i] = ??  
22             alphabetCnt += 1;  
23         }  
24     }  
25     printf("%s", str);  
26     printf("count of alphabet: %d", alphabetCnt);
```

Outline

- Review
- Array: Showcase, String
- **Array: Showcase, 2D**
- Assignment



Array: Showcase, 2D

2D array declaration, and element index

```
3  #define MAX_ROW 16
4  #define MAX_COL 64
5
6  int main()
7  {
8      int matrix[MAX_ROW][MAX_COL] = {0};
9      for (int i = 0; i < MAX_ROW; i++) {
10         for (int j = 0; j < MAX_COL; j++) {
11             matrix[i][j] = 100 * i + j;
12         }
13     }
```


Array: Showcase, 2D

2D array still a continuous memory space!

```
14     printf("address of matrix: %p \n", matrix);
15     for (int i = 0; i < MAX_ROW; i++) {
16         printf("matrix[%d]: %x \n", i, matrix[i]);
17         printf("matrix[%d][0]: pointer %p, value %d\n",
18             i, &matrix[i][0], matrix[i][0]);
19     }
20 }
```

address of matrix: 000000000061EE10

matrix[0]: ??

matrix[0][0]: pointer 0000000000 ?? value 0

matrix[1]: ??

matrix[1][0]: pointer 0000000000 ?? value 100

Array: Showcase, 2D

2D array still a continuous memory space!

```
14     printf("address of matrix: %p \n", matrix);
15     for (int i = 0; i < MAX_ROW; i++) {
16         printf("matrix[%d]: %x \n", i, matrix[i]);
17         printf("matrix[%d][0]: pointer %p, value %d\n",
18             i, &matrix[i][0], matrix[i][0]);
19     }
20 }
```

```
address of matrix: 000000000061EE10
```

```
matrix[2]: ??
```

```
matrix[2][0]: pointer 0000000000 ??, value 200
```


Array: Showcase, 2D

Matrix Transpose (矩阵转置)

```
8      int matrix[MAX_ROW][MAX_COL] = {0};
9      int row, col;
10
11     printf("plz input matrix row and column: ");
12     scanf("%d %d", &row, &col);
13
14     printf("plz input matrix values, row by row: \n");
15     for (int i = 0; i < row; i++) {
16         for (int j = 0; j < col; j++) {
17             scanf("%d", &matrix[i][j]);
18         }
19     }
```

Array: Showcase,

Matrix Transpose (矩阵转置)

```
21     int transRow = col;
22     int transCol = row;
23     int transMat[MAX_ROW][MAX_COL] = {0};
24     for (int i = 0; i < transRow; i++) {
25         for (int j = 0; j < transCol; j++) {
                ??
27         }
28     }
29
30     puts("Transposed Matrix: ");
31     for (int i = 0; i < transRow; i++) {
32         for (int j = 0; j < transCol; j++) {
33             printf("%d\t", transMat[i][j]);
34             // printf("%d\t", matrix[i][j]);
35         }
36         printf("\n");
37     }
38 }
```

Outline

- Review
- Array: Showcase, String
- Array: Showcase, 2D
- **Assignment**



Assignment 1)

回文检测

A palindrome is a word, phrase, number, or other sequence of characters that reads the same backward as forward, ignoring spaces, punctuation, and capitalization. Your task is to write a program that checks if a given sentence is a palindrome.

Note that your program must be able to handle sentences of varying lengths and should be efficient in its execution.

Input

A single line containing the sentence to be checked. The sentence consists of less than 10^6 characters.

Output

Print **Yes** if the sentence is a palindrome. Print **No** if it is not.

```
A man a plan a canal Panama
Yes
```

Assignment 2)

矩阵乘法

For matrix M , let $M_{i,j}$ be the entry on the i^{th} row and j^{th} column. Both row index and column index start from 0.

Given two matrices A and B , please output a specific sub-matrix of C , where $C = A \times B$.

- A has m rows and n columns.
- B has n rows and p columns.
- The result C has m rows and p columns.

Recall matrix multiplication, we have:

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

Please output the sub-matrix $C_{i,j}$ ($x_1 \leq i \leq x_2, y_1 \leq j \leq y_2$), where x_1, x_2, y_1, y_2 are given by the input.

Assignment 2)

矩阵乘法

Input

The first line consists of three integers m, n, p ($1 \leq m, n, p \leq 1000$).

For the following m lines, each line consists of n integers, representing matrix A ($|A_{i,j}| \leq 10^6$).

For the following n lines, each line consists of p integers, representing matrix B ($|B_{i,j}| \leq 10^6$).

The last line consists of four integers x_1, x_2, y_1, y_2 , where $0 \leq x_1 \leq x_2 < m$, $0 \leq y_1 \leq y_2 < p$, and $(x_2 - x_1 + 1) \times (y_2 - y_1 + 1) \leq 2000$, meaning that the target sub-matrix contains no more than 2000 entries.

Output

Print the specified sub-matrix of C . The result consists of $x_2 - x_1 + 1$ lines, and each line contains $y_2 - y_1 + 1$ integers.

```
2 2 2
1 2
3 4
1 0
0 1
0 1 0 1
1 2
3 4
```

THANK YOU

