Arrays and Strings

存储同类型的多个元素

Store multi elements of the same type

数组(array)

- 存储固定数目的同类型元素。
  如整型数组存储的是一组整数，字符数组存储的是一组字符。数组的大小称为数组的尺度(*dimension*).
- 定义格式：

  type arrayName[dimension];

  如声明4个元素的整型数组： int arr[4];

- 可用索引(index)访问数组的元素，索引(也称为下标)从 0开始编号，如访问arr的第三个元素可通过arr[2]。

  arr[2] = 20;

数组(array)-应初始化

- 否则会导致不可预期的逻辑错误。
- 初始化有多种方式：
  - **初始化方式1**：先声明数组，然后初始化其中的元素。如：

```
int arr[4];
arr[0] = 6;
arr[1] = 0;
arr[2] = 9;
arr[3] = 6;
```

6	0	9	6

数组(array)-应初始化

- 初始化方式**2**：声明时初始化其中的部分或全部元素。如：
```
int  arr[4] = { 6, 0, 9, 6 };
```
这种方式初始化全部元素时，有时省略数组个数的说明，让编译器帮我们推断数组的大小。如：
```
int arr[] = { 6, 0, 9, 6, 2, 0, 1, 1 };
```
- 访问数组元素时，其索引是0到n-1的整数(n是数组的大小)或整型表达式，否则会引起程序崩溃。
```
int arr[] = { 6, 0, 9, 6, 2, 0, 1, 1 };
int i =2;
arr[5];
arr[i];
arr[i+3];
```

数组(array)-应初始化

```
#include <iostream>
using namespace std;

int main() {

    int arr[4];
    cout << "Please enter 4 integers:" << endl;

    for(int i = 0; i < 4; i++)
          cin >> arr[i];

    cout << "Values in array are now:";

    for(int i = 0; i < 4; i++)
          cout << " " << arr[i];

    cout << endl;

    return 0;
}
```

数组(array)-作为函数参数

- 数组可以作为一个参数传给函数。声明函数时，只要指明该参数是一个(但不带大小)数组。如

```
#include <iostream>
using namespace std;

int sum(const int array[], const int length) {
    long sum = 0;
    for(int i = 0; i < length; sum += array[i++]);
    return sum;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5, 6, 7};
    cout << "Sum: " << sum(arr, 7) << endl;
    return 0;
}
```

数组(array)-作为函数参数

- 数组是作为引用(reference)被传递给函数的，因此在函数内对数组的修改也是对实际参数的修改。

```
void f(int arr[]){
    arr[2] = 33;
}
int main(){
   int a[] = {1,2,3,4,5};
   f(a);
   for(int i = 0 ; i<5 ;i++)
      std::cout<< a[i]<<” ”;
   return 0;
}
```

多维数组(multidimensional array)

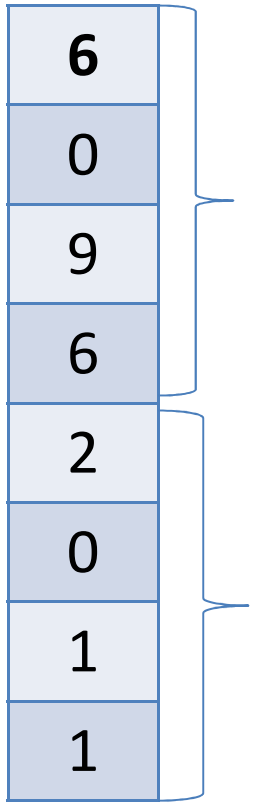
- 如二维数组：

  ```
  type arrayName[dimension1][dimension2];
  ```

  数组将包含*dimension1*dimension2*个元素.第一索引将指示*dimension1*个子数组,第二索引将指示*dimension2*个子数组。
- 初始化和访问方式类似于一维数组。

多维数组(multidimensional array)

```
int main() {
    int twoDimArray[2][4];
    twoDimArray[0][0] = 6;
    twoDimArray[0][1] = 0;
    twoDimArray[0][2] = 9;
    twoDimArray[0][3] = 6;
    twoDimArray[1][0] = 2;
    twoDimArray[1][1] = 0;
    twoDimArray[1][2] = 1;
    twoDimArray[1][3] = 1;

    for(int i = 0; i < 2; i++)
        for(int j = 0; j < 4; j++)
            cout << twoDimArray[i][j];

    cout << endl;
    return 0;
}
```



多维数组(multidimensional array)

- 也可如下方式初始化数组：

```
int twoDimArray[2][4] = { 6, 0, 9, 6, 2, 0, 1, 1 };
int twoDimArray[2][4] = { { 6, 0, 9, 6 } , { 2, 0, 1, 1 } };
```

- 声明多维数组必须指定每一维的大小，作为函数参数时，除第一维外，必须指定其他维的大小。

```
int aFunction(int arr[][4]) { … }
```

- 多维数组仅仅是程序员的抽象，实际上在内存内仍然是和一维数组一样：是存放在内存的一个序列。因此

```
int arr[2][4];      和      int arr[8];
```

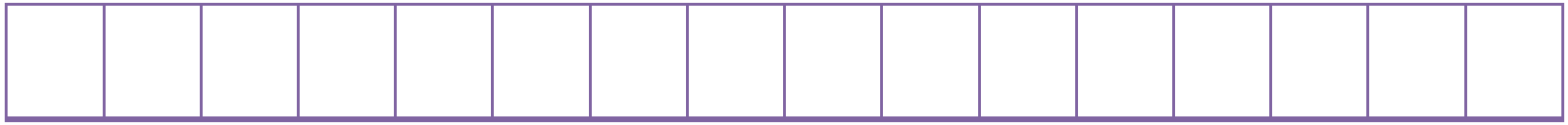
在内存中是同一个东西！

字符串(Strings)

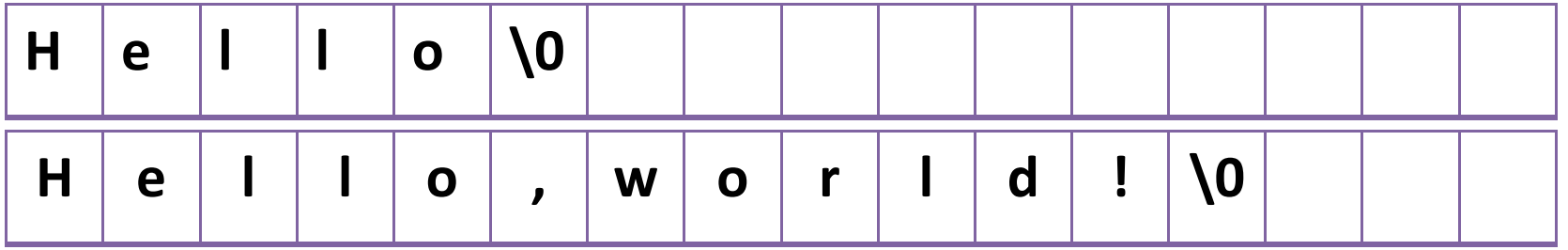
- 字符串实际就是一串连续的字符序列，所以我们也可以用简单的字符数组来表示它。

  例如：char jenny [16];

- 表示一个最多可以存储16个字符的数组。



- 它也可以存储比这短的字符序列，而且实际中常常如此。一种习惯在字符串的有效内容的结尾处加一个空字符(null character)来表示字符结束，它的常量表示可写为0 或'\0'



字符串(Strings)

- C语言字符串：以空字符 ‘\0’结束的字符数组。
- 两种初始化方式：

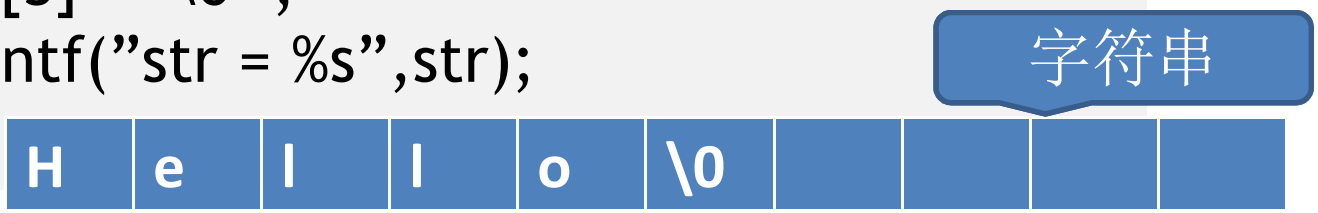
```
char mystring [ ] = { 'H', 'e', 'l', 'l', 'o', '\0' };
char mystring [ ] = "Hello";
```

```
#include <iostream>
using namespace std;

int main() {
   char helloworld[] = {   'H', 'e', 'l', 'l', 'o', ',', ' ',
                           'w', 'o', 'r', 'l', 'd', '!', '\0' };

   cout << helloworld << endl;

   return 0;
}
```

字符串(Strings)

- 字符串：以空字符 '\0'结束的字符数组

```
#Include <stdio.h>
int main(){
    char str[10];
    str[0] = 'H' ;  str[1] = 'e' ; str[2] = 'l' ;
    str[3] = 'l' ;   str[4] = 'o' ;
    printf("str = %s",str);
    str[5] = '\0' ;
    printf("str = %s",str);
}
```



- 字符串可以用文字常量初始化：

```
int helloworld[] = "Hello, world!";
```

编译器会自动在最后插入空字符

C/C++标准库中的字符串处理函数

- C标准库提供了不同的字符串处理函数，这些函数的原型在下列文件中说明了。
  - cctype (ctype.h): character handling
  - cstdio (stdio.h): input/output operations
  - cstdlib (stdlib.h): general utilities
  - cstring (string.h): string manipulation

- C++标准库提供了cstring库取代string.h

```
#include <iostream>
#include <cctype>
using namespace std;
int main() {
    char str[ ] = "t6H0I9s6.iS.999a9.STRING";

    char ch = str[0];
    for(int i = 0; ch != '\0'; ch = str[++i]) {
       if(isalpha(ch))
          cout << (char)(isupper(ch) ? tolower(ch) : ch);
       else if(ispunct(ch))
       cout << ' ';
    }

    cout << endl;
    return 0;
}
```

用cstring取代string.h

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {
    char fragment1[] = "I'm a s";
    char fragment2[] = "tring!";
    char fragment3[20];
    char finalString[20] = "";

    strcpy(fragment3, fragment1);
    strcat(finalString, fragment3);
    strcat(finalString, fragment2);

    cout << finalString;
    return 0;
}
```

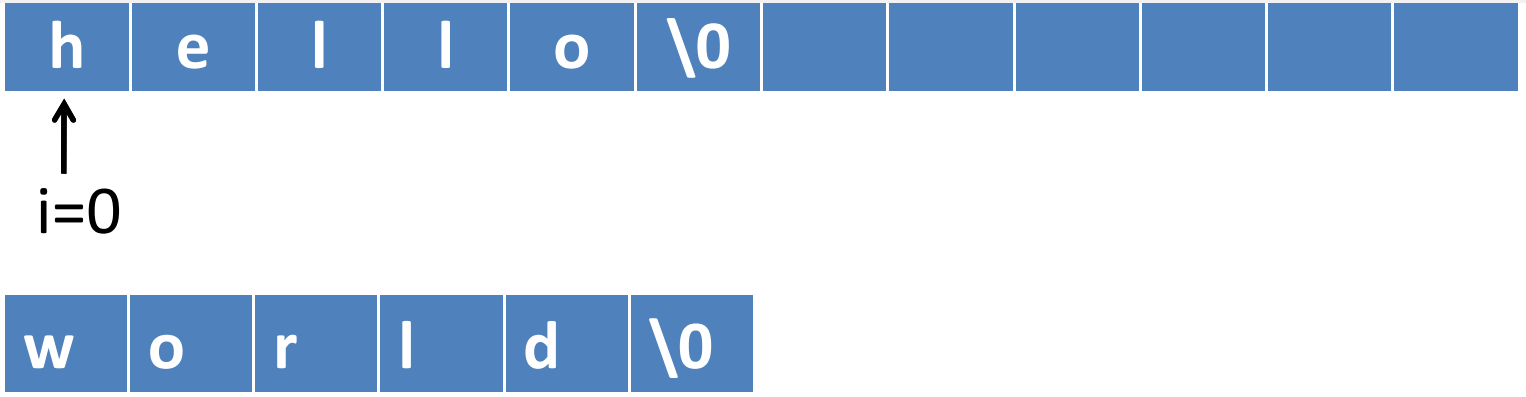
cstring/string.h的函数实现模拟

```
/*求字符串长度*/
int strlen(char str[]){
   int i = 0 ;
   while (str[i]!= '\0') i++;
   return i;
}
```

```
/*字符串拼接*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= '\0') i++;
   for(int j = 0 ; str2[j]!= '\0' ;j++,i++) str[i] = str2[j];
    str[i] = '\0;
   return ;
}
```

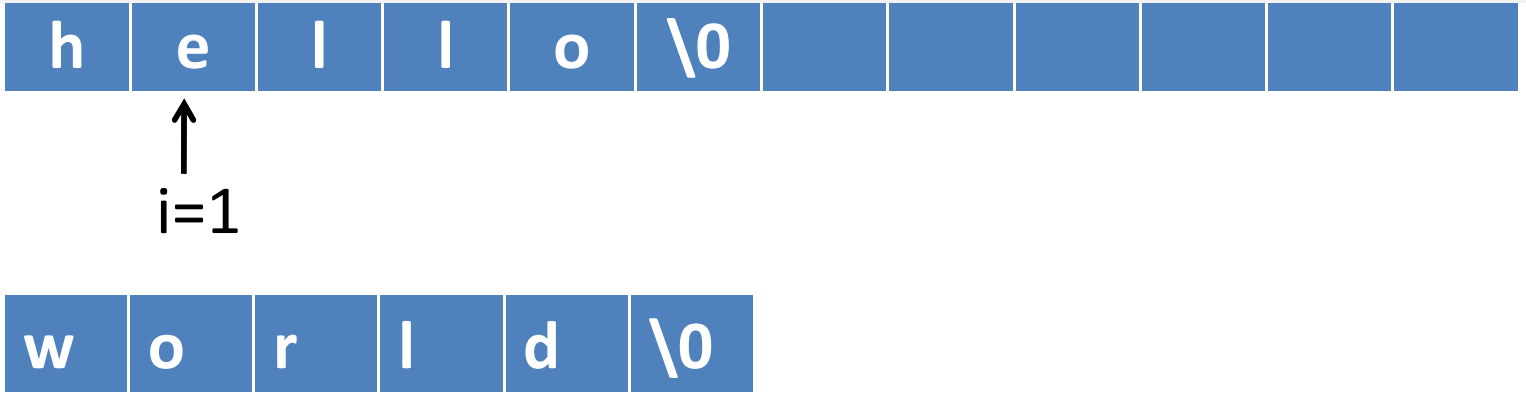
cstring/string.h的函数实现模拟

```
/*字符串拼接*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= '\0') i++;
   for(int j = 0 ; str2[j]!= '\0' ;j++,i++) str[i] = str2[j];
    str[i] = '\0;
   return ;
}
```



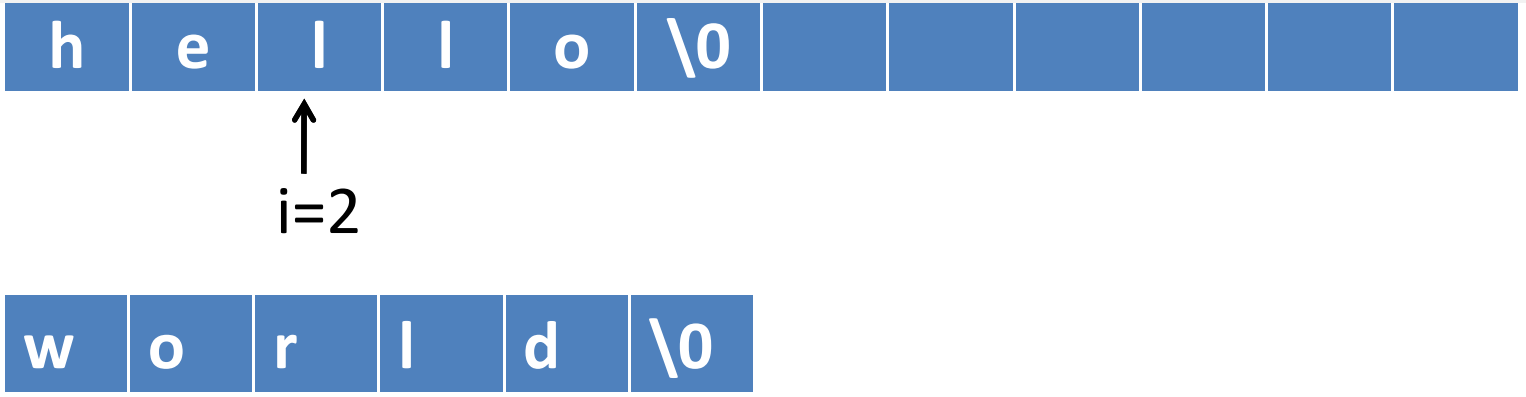
cstring/string.h的函数实现模拟

```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= ’\0’) i++;
   for(int j = 0 ; str2[j]!= ’\0’ ;j++,i++) str[i] = str2[j];
    str[i] = ’\0;
   return ;
}
```



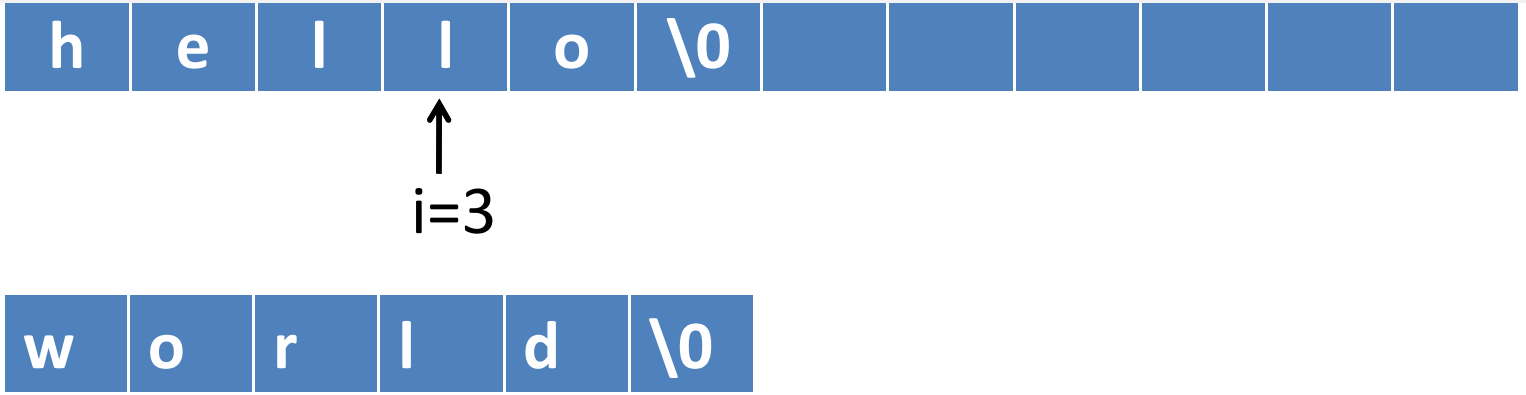
cstring/string.h的函数实现模拟

```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= ’\0’) i++;
   for(int j = 0 ; str2[j]!= ’\0’ ;j++,i++) str[i] = str2[j];
    str[i] = ’\0;
   return ;
}
```



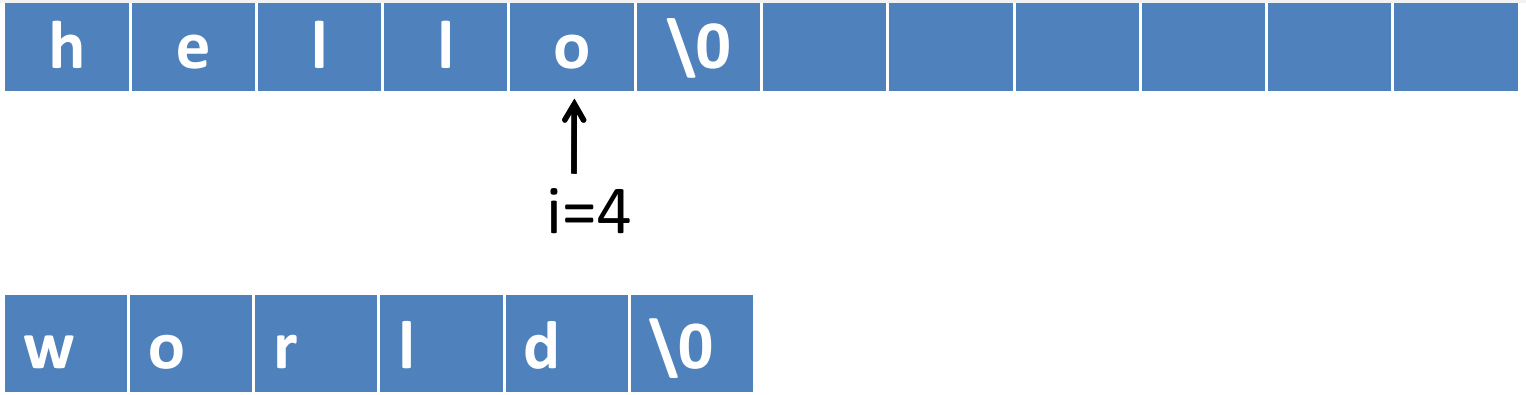
cstring/string.h的函数实现模拟

```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= ’\0’) i++;
   for(int j = 0 ; str2[j]!= ’\0’ ;j++,i++) str[i] = str2[j];
    str[i] = ’\0;
   return ;
}
```



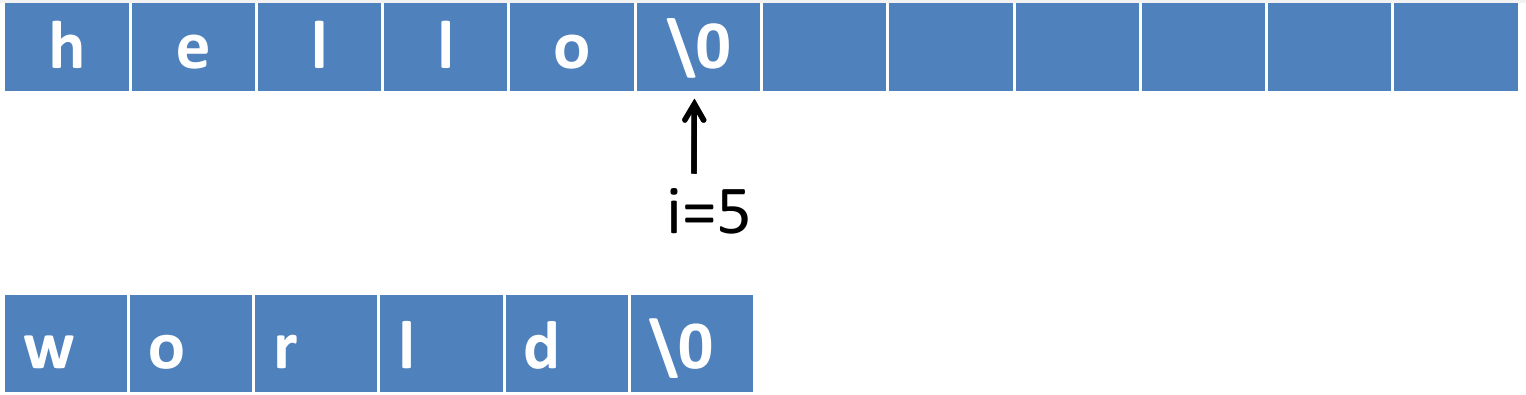
cstring/string.h的函数实现模拟

```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= ’\0’) i++;
   for(int j = 0 ; str2[j]!= ’\0’ ;j++,i++) str[i] = str2[j];
    str[i] = ’\0;
   return ;
}
```



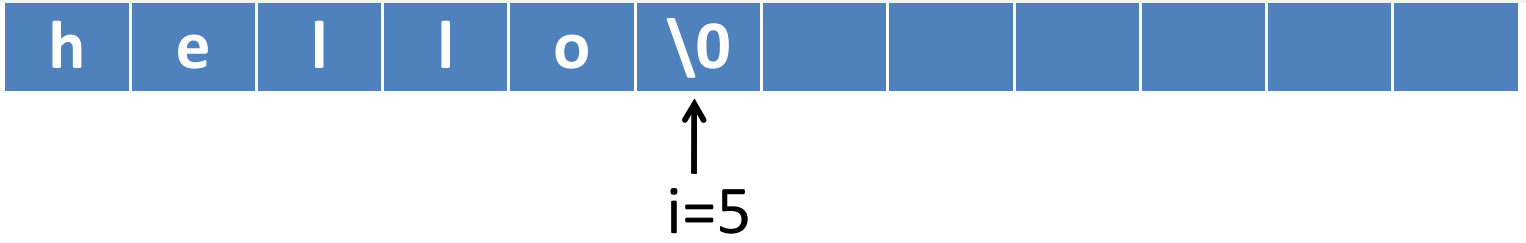
cstring/string.h的函数实现模拟

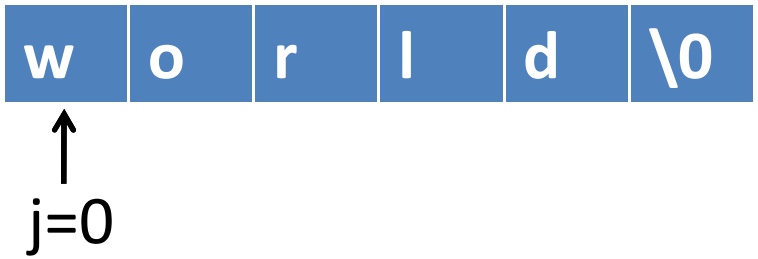
```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= '\0') i++;
   for(int j = 0 ; str2[j]!= '\0' ;j++,i++) str[i] = str2[j];
    str[i] = '\0;
   return ;
}
```



cstring/string.h的函数实现模拟

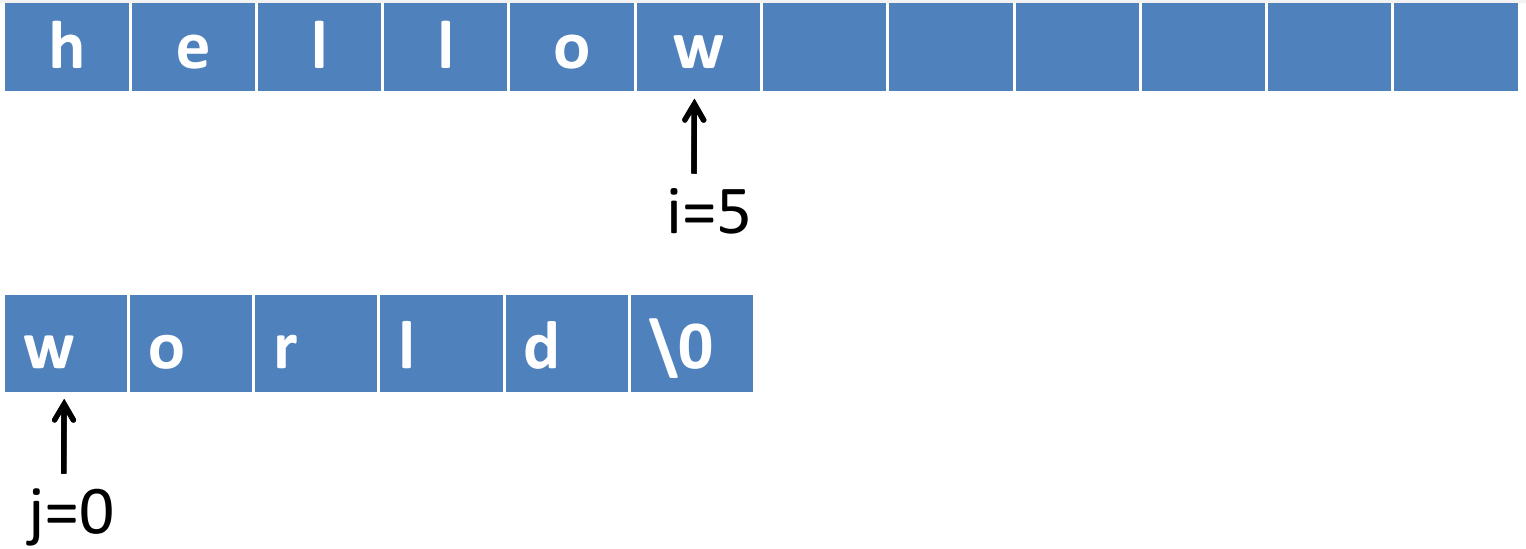
```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= ’\0’) i++;
   for(int j = 0 ; str2[j]!= ’\0’ ;j++,i++) str[i] = str2[j];
    str[i] = ’\0;
   return ;
}
```





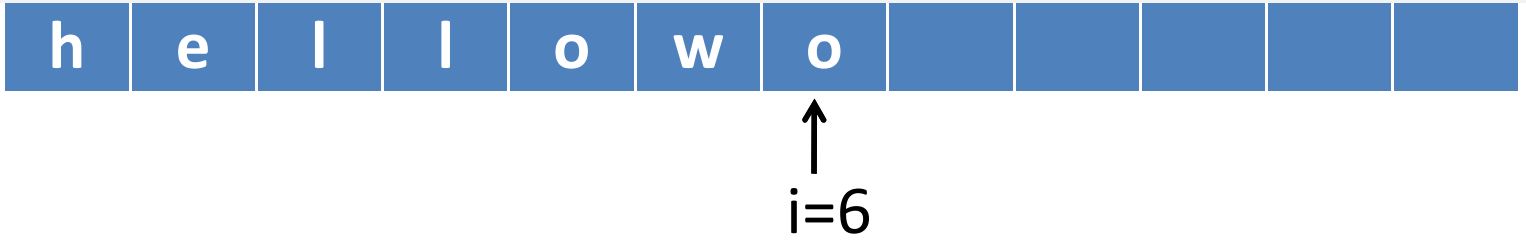
cstring/string.h的函数实现模拟

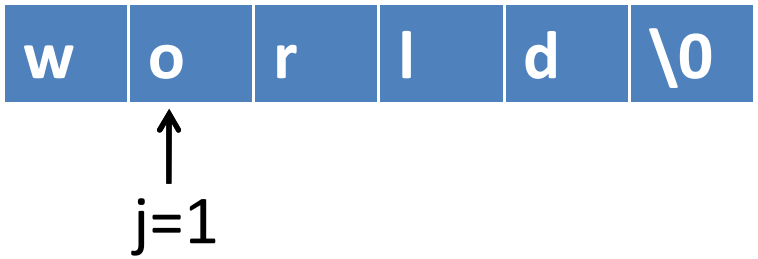
```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= '\0') i++;
   for(int j = 0 ; str2[j]!= '\0' ;j++,i++) str[i] = str2[j];
    str[i] = '\0;
   return ;
}
```



cstring/string.h的函数实现模拟

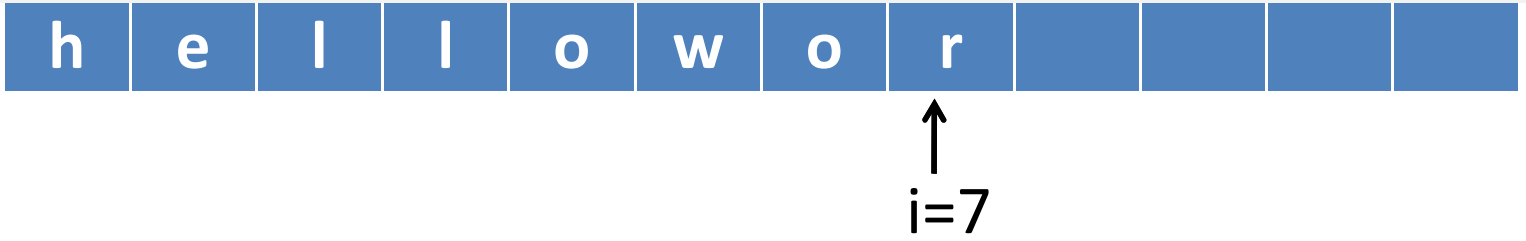
```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= ’\0’) i++;
   for(int j = 0 ; str2[j]!= ’\0’ ;j++,i++) str[i] = str2[j];
    str[i] = ’\0;
   return ;
}
```

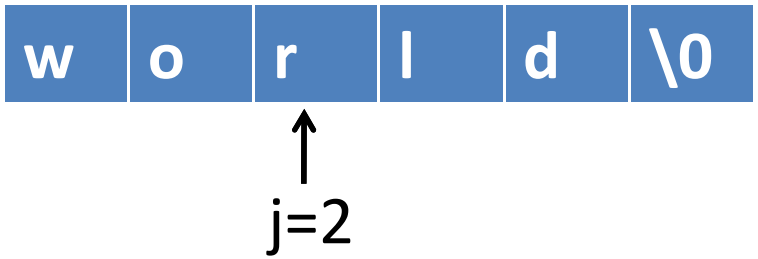




cstring/string.h的函数实现模拟

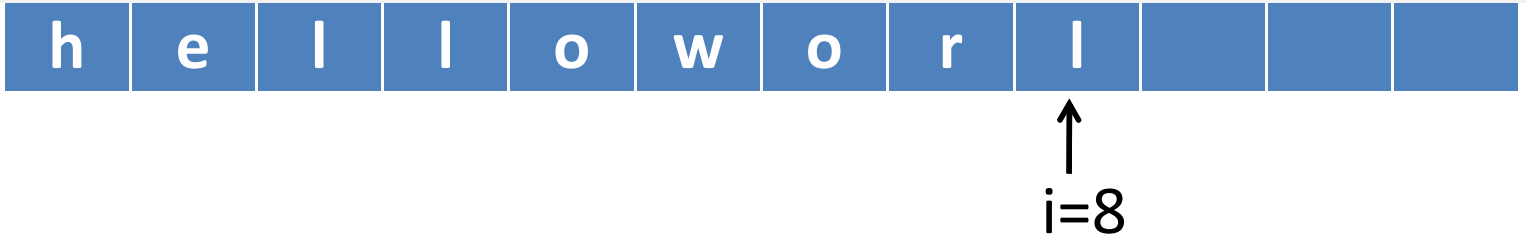
```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= ’\0’) i++;
   for(int j = 0 ; str2[j]!= ’\0’ ;j++,i++) str[i] = str2[j];
    str[i] = ’\0;
   return ;
}
```

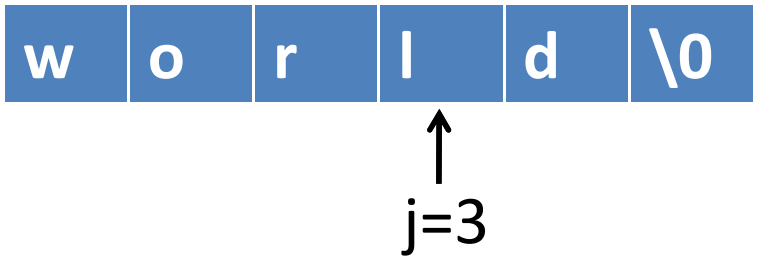




cstring/string.h的函数实现模拟

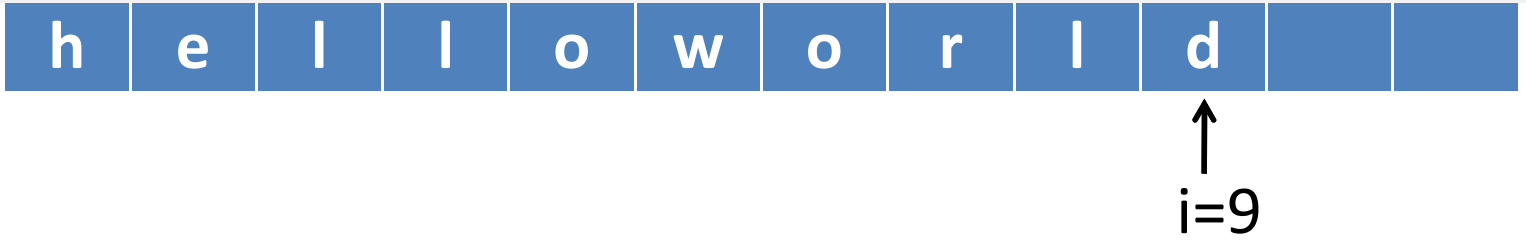
```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= ’\0’) i++;
   for(int j = 0 ; str2[j]!= ’\0’ ;j++,i++) str[i] = str2[j];
    str[i] = ’\0;
   return ;
}
```

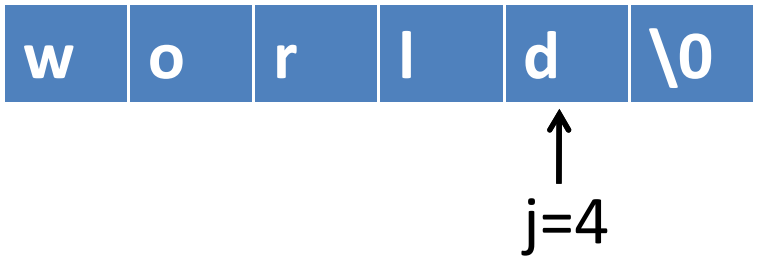




cstring/string.h的函数实现模拟

```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= ’\0’) i++;
   for(int j = 0 ; str2[j]!= ’\0’ ;j++,i++) str[i] = str2[j];
    str[i] = ’\0;
   return ;
}
```





cstring/string.h的函数实现模拟

```
/*求字符串长度*/
void strcat(char str[],char str2[]){
   int i = 0 ;
   while (str[i]!= ’\0’) i++;
   for(int j = 0 ; str2[j]!= ’\0’ ;j++,i++) str[i] = str2[j];
    str[i] = ’\0;
   return ;
}
```