指针和数组

2022年3月19日 9:51

1. 数组名作为指针.

数组名可以作为指向索引为0元素的指针.

```
int main(void) {
    int arr[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    *arr = 100;
    *(arr + 7) = 700;

    // arr++; 数组作为指针时是一个指针常量
```

把数组名作为指针使用时，可以简化我们的代码.

```
int arr[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
int sum = 0;

for (int* p = arr; p < arr + 10; p++) {
    sum += *p;
}
printf("sum = %d\n", sum);
```

2. 指针也可以作为数组名来使用 (可以对指针使用[]运算符)

p[i] = *(p+i)

```
int find_largest(int* arr, int n) {
    int max = arr[0];
    for (int i = 0; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }

    //int max = *arr; // p[0] 等价于 *(p + 0)
    //for (int i = 0; i < n; i++) {
    //  if (*(arr + i) > max) {
    //      max = *(arr + i);
    //  }
    //}

    return max;
}
```

总结，指针和数组之间的关系.

① 可以利用指针处理数组 (指针的算术运算)

② 数组名可以作指向该数组索引为0元素的指针

③ 指针也可以作为数组名 (可以对指针使用[]运算符. p[i] = *(p+i))

字符串

2022年3月19日 10:12

#1. 字符串字面量

用双引号括起来的字符序列就是字符串字面量

```
printf("I love xixi!\n");
printf("I love xixi!"
    "  --From Thomas He");
```

编译器会把两个相邻的字符串字面量合并成一个字符串字面量!

相邻：当两个字符串字面量仅以空白字符分割时，我们就认为它们是相邻的.

(1) 字符串字面量是如何存储的

C语言会用字符数组来存储字符串字面量. "abc" → |a|b|c|\0|

"" → |\0|

printf("Hello kity.\n")

↳ 这里传递其实是一个字符指针 char*.

#2. 字符串变量

C语言没有专门的字符串类型，C语言的字符串依赖字符数组存在.

```
char name[10] = "Allen";
```

字符串 → |A|l|l|e|n|\0|\0|\0|\0|\0|

0 1 2 3 4 5 6 7 8 9

↓

是字符串字面量，本质是一个初始化式 {'A','l','l','e','n'} 的简化形式.

(1) 字符串的初始化

char name[10] = "Allen";

char name[5] = "Allen"; |A|l|l|e|n| 这不是一个字符串.

```
char name[5] = "Allen";
printf("%s", name);
```

Microsoft Visual Studio 调试控制台

Allen烫烫烫(?

char name[] = "Allen"; |A|l|l|e|n|\0| 推荐做法

(2) 字符串的初始化和字符指针的初始化.

char name[] = "Allen"; name |A|l|l|e|n|\0| (可以修改)

↳ 数组的初始化式

char * name = "Allen";

↳ 字符串字面量

name → |A|l|l|e|n|\0| (常量，不能够修改)

```
char name1[] = "Allen";
char* name2 = "Allen";
name1[0] = 'a';
name2[0] = 'a';

return 0;
```

已引发异常

引发了异常: 写入访问权限冲突。
name2 是 0xCA7B30。

#3. 读写字符串.

写. printf + %s

puts
↳string

```
char name[] = "Beyonce";
printf("%s\n", name);
puts(name);
```

```
Microsoft Visual Studio 调试控制台
Beyonce
Beyonce
```

→ puts 写完字符串后，会在后面添加额外的换行符

puts (name) 等价于 printf("%s\n", name)

读. scanf + %s (一般不使scanf) (X)

gets (X)

```
char name[N];
scanf("%s", name);
```

scanf: 会跳过前面的空白字符，读取字符存入到数组，直到遇到空白字符为止. 然后会在后面添加'\0'

注意事项. ① 永远不会包含空白字符.

② scanf不会检查数组越界

```
char name[N];
// scanf("%s", name);
gets(name);
```

gets: 不会跳过前面的空白字符，读取字符存入到数组，直到遇到换行符为止. 然后在后面添加'\0'

注意事项. gets不会检查数组是否越界.

```
char *gets_s( char *str, rsize_t n );
```
→ n 表示数组的长度，所以最多能够读 n-1 个字符.

作业：利用 getchar() 实现类似 gets_s 的效果. read_line.

#4. 字符串的库函数.

(指明在 strlen 中不会修改 s 指向的内容，传入参数)

① size_t strlen (const char *s);

计算字符串的长度，不包括'\0'

```
printf("strlen(\"abc\") = %d\n", strlen("abc"));
printf("strlen(\"\") = %d\n", strlen(""));
```

```
Microsoft Visual Studio 调试控制台
strlen("abc") = 3
strlen("") = 0
```

② int strcmp(const char *s1, const char *s2);

按字典顺序比较 s_1 和 s_2 → "abc" < "cba", "abc" < "abd", "abc" < "abcd"

如果 $s_1 > s_2$，则返回正整.

如果 $s_1 = s_2$，则返回零

如果 $s_1 < s_2$，则返回负整

③ char *strcpy(char *s1, const char *s2)

↳ 表明通常会在函数中会修改 s1 指向的内容，传入传出参数

把 s_2 指向的字符串复制到 s_1 指向的数组中.

Char $s_1[4]$;

strcpy (s_1, "Hello"), s_1 | H | e | l | l | 0 | \0 | (不会检查数组是否越界).

Defined in header <string.h>

```
char *strncpy( char *dest, const char *src, size_t count );
```

↳ 写入字符的个数

```
char s1[10];
strncpy(s1, "Hello", 4);
strncpy(s1, "Hello", 6);
strncpy(s1, "Hello", 8);
```

s_1 | H | e | l | l | 不会写入空字符 \0

s_1 | H | e | l | l | l | 0 | \0 |

s_1 | H | e | l | l | 0 | 0 | 0 | 0 | \0 | 会在后面添加额外的字符, 直到写入8个字符.