字符数组函数，进制转换，指针:

Due on Nov. 25, 2023

风之树

目录

.1 字符数组函数

.1.1 字符数组函数

- 字符串的复制.

  函数名: strcpy(str1,str2)
  头文件: <cstring>
  功能: 将str2的内容复制到str1中去 要求str1的length比str2的length大

- 字符串的交换

  函数名: swap(str1,str2)
  头文件: <algorithm>
  功能: 交换两个字符串的内容，要求二者的有效字符串长度小于其总长度

- 字符串的比较

  函数名: strcmp(str1,str2)
  头文件: <cstring>
  功能: 按照字典顺序比较str1和str2，如果str1大于str2，那么返回一个1，

  如果小于那么返回-1,如果相等返回0

- 字符串的连接

  函数名: strcat(str1,str2)
  头文件:<cstring>
  功能: 将str2拼接到str1的右边，要求str1有足够的空间容纳str2

- 字符串的子串

  函数名: strstr(str1,str2)
  头文件: <cstring>
  功能: 判断str2是否是str1的子串，如果不是返回NULL，如果是返回出现的首次地址

- 字符串的长度

  函数名: strlen(str1)
  头文件: <cstring>
  功能: 返回字符串的长度

- 将字符转换为大小写

函数名：toupper(ch), tolower(ch)
头文件：<ctype>
功能：将大写字符/小写字符转换为 小写字符或者大写字符

-

.1.2 二维字符串的排序

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int M = 5;  // 假设 M 为字符串数组的行数
const int N = 20; // 假设 N 为字符串数组中每个字符串的最大长度

// 交换函数的定义，你也可以使用<iostream>中的函数
void swap(char a[], char b[]) {
    char temp[N + 1];
    strcpy(temp, a);
    strcpy(a, b);
    strcpy(b, temp);
}

// 选择排序

void selectionSort(char word[M][N + 1]) {
    for (int i = 0; i < M - 1; ++i) {
        int minIndex = i;
        for (int j = i + 1; j < M; ++j) {
            if (strcmp(word[j], word[minIndex]) < 0) {
                minIndex = j;
            }
        }
        if (minIndex != i) {
            swap(word[i], word[minIndex]);
        }
    }
}

int main() {
    char word[M][N + 1] = {"apple", "orange", "banana", "grape", "kiwi"};

    cout << "Before sorting:" <<  endl;
    for (int i = 0; i < M; ++i) {
         cout << word[i] <<  endl;
    }

    sort(word,word+M);

    cout << "\nAfter sorting:" <<  endl;
    for (int i = 0; i < M; ++i) {
```

```
44          cout << word[i] << endl;
45      }
46
47  }
```

.2 进制转换

.2.1 计算机常用的进制

- 二进制 Binary B
- 八进制 Octal O
- 十进制 Decimal D
- 十六进制 Hexadecimal H

.2.2 位值定理

若一个数的进制为 b，并且这个数字为 $d_n d_{n-1} ... d_0$，那么这个数的十进制为 $n = d_n \times b^n + d_{n-1} \times b^{n-1} + ... + d_1 \times b^1 + d_0 \times b^0$

.2.3 R 进制转换为十进制

根据位值定理

```
#include<iostream>
#include<cstring>
using namespace std;
const int N = 50;
char s[N];
int convert_to_decimal(char s[],int r);
int main()
{
    int r;
    cin >> s >> r;
    int ans = convert_to_decimal(s,r);
    cout << ans;
    return 0;
}
int convert_to_decimal(char s[], int r)
{
    int n =0;
    int w =1;
    for(int i =strlen(s)-1; i>=0 ;i--) // 关键是循环的初始条件，结束条件，以及变量更新
    {
        if(s[i] >= '0's[i]<='9')
        {
            n += (s[i]-'0') * w;
        }
        else
        {
            n += (s[i]-'A'+10)*w;
        }
        w *= r;
    }
    return n;
}
```

.2.4 十进制转换为 R 进制

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

string decimalToRBase(int number, int base) {
    string result = "";

    while (number > 0) {
        int remainder = number % base; // 取余数
        char digit;

        if (remainder < 10) {
            digit = remainder + '0'; // 0-9的数字直接转为字符
        } else {
            digit = remainder - 10 + 'A'; // 大于9的数字转为对应的字母(A-F)
        }

        result = digit + result; // 将每一位余数加到结果的最前面
        number /= base; // 更新被除数为商
    }

    return result;
}

int main() {
    int decimalNumber = 217; // 要转换的十进制数
    int rBase = 16; // 转换的目标进制，这里选择16进制

    string result = decimalToRBase(decimalNumber, rBase);

    cout << "Decimal number " << decimalNumber << " in base " << rBase << " is: " << result << endl;

    return 0;
}
```

.3 指针

.3.1 指针介绍

- 指针是一种数据类型，可以访问内存地址
- 指针可以方便的处理字符串和数组
- 指针很危险，稍有不慎，就会造成内存的泄漏，程序的不稳定以及系统崩溃
- 指针是学习 C++ 中很重要的一环，是用来衡量是否掌握 C++ 语言的一个标志

.3.2 变量在内存中的存储

- 计算机内存中给一个变量分配给一定的内存空间
- 内存给每个字节都有唯一的编号，即地址

1

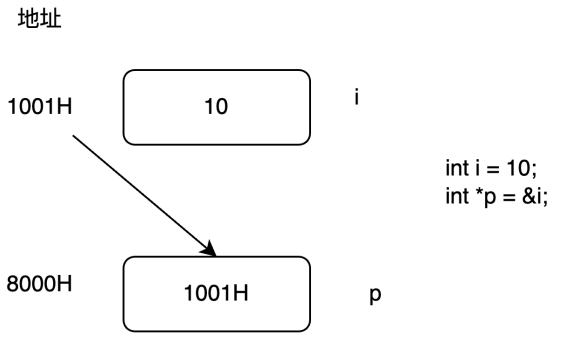

图 1: 指针示意图

.3.3 指针变量

- 指针变量是指专门存储其他变量地址的变量，这个变量叫做指针变量
- 指针也占用一定的内存，存储的是其他变量的地址
- 指针变量定义的语法为

```
类型标识符 *指针变量名
int *a;
double *b;
分别表示一个指向int类型的指针变量a一个指向double类型的指针变量b
```

- 指针变量的初始化

```
(1) int *a = NULL    // 将指针初始化为空指针
(2) int i; int *p = &i;  //将指针指向i所占的内存
(3) int *p = new(int);   //指针指向一个新开辟的内存空间，该内存空间指向int类型
```

.3.4 指针和数组

- 数组在内存中占用着连续的内存单元
- 数组名是连续内存单元的首地址，也是第一个元素的首地址
- 指针变量可以指向数组，也可以指向数组元素
- 指针变量虽然是内存地址，但是也可以进行加减操作，一般是配合数组操作的 p++ 不是指 p 的值 +1，而是根据 p 指向的数据类型所占的空间大小增加，即正好跳过一个元素所占的空间，例如如果 p 是 int 类型的指针，p++ 实际上地址增加了 sizeof(int) 的空间
- 通过指针引用数组元素

  (1) 下标法：使用p[i] a[i]访问数组元素
  (2) 指针法：即使用*(p+1) 和*(a+1) 访问数组元素a[i]

.3.5 指针与函数

函数的参数可以是指针类型，他的作用是将变量的一个地址传送到函数中将数组名称作为函数参数传递，实际上是指将数组的首地址传递给了函数，这样可以访问数组中的任意一个元素

.3.6 基于指针的交换函数

```cpp
#include <iostream>

using namespace std;
void swap(int *ptr1, int *ptr2) {
    int temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

int main() {
    int num1 = 5;
    int num2 = 10;

    cout << "Before swapping: num1 = " << num1 << ", num2 = " << num2 << endl;

    // 传递变量地址给 swap 函数
    swap(num1, num2);

    cout << "After swapping: num1 = " << num1 << ", num2 = " << num2 << endl;

    return 0;
}
```