作业讲解

2024年4月25日 9:21

(a) double ans = 10.0 + 2.0 / (3.0 - 2.0) * 2.0;请在适当的位置插入()，使得 ans = 11.0.

(b) double ans = 18.0 / SQUARED(2 + 1); 如果宏函数定义如下，请写出对应 ans 的值。

```
#define SQUARED(x) x*x
#define SQUARED(x) (x*x)
#define SQUARED(x) (x)*(x)
#define SQUARED(x) ((x)*(x))
```

① 18.0 / 2+1 * 2+1 = 9 +2 +1 = 12.0

② 18.0/(2+1 * 2 +1) = 18.0 / 5 = 3.6

③ 18.0 / (2+1) * (2+1) = 18.0

④ 18.0 / ((2+1) *(2+1)) = 2.0

(c) 请解释下面代码为什么错了，并改正它。

```
1) #include <stdio.h>;
2) int function(void arg1){
       return arg1-1;
   }
3) #define MESSAGE = "Happy new year!"
```

int

假设 int n = 0xCAFE 请用表达式完成下面操作 (拓展题：不要求每个同学都写)

(a) 测试最后 4 位中是不是最少有 3 位为 1.

(b) 逆转字节序(i.e.,使 n = 0xFECA)

(c) 旋转 4 位 (i.e., 使 n = 0xECAF)

左移

hello.java → (javac) → 字节码

CAFE BABE

(a) int x = n & 0xF;

0111 1011 1101 1110 1111

7 B D E F

x == 7 || x == B || x == D

(b) CA FE → FE CA

n >> 8 : 00 CA

(n & 0xFF) << 8 FE 00

(n >> 8) | ((n & 0xFF) << 8)

(c) CAFE → ECAF

n >> 4 : 0CAF

(n & 0xF) << 12 : E000 (1)

利用优先级规则，计算下面表达式的值，并确定各个变量的值(不运行代码)。添加括号，显示表示优先级关系。

(a) 假设(x = 0xFF33, MASK = 0xFF00).表达式: c = x & MASK ==0;

(b) 假设(x = 10, y = 2, z = 2;).表达式: z = y = x++ + ++y * 2;

(c) 假设(x = 10, y=4, z=1;).表达式: y >>= x & 0x2 && z;

0

(a) (c = (x & (MASK == 0)))

x = 0xFF33, MASK = 0xFF00, c = 0

(b) (z = (y = ((x++) + ((++y) * 2))))

z = (y = (10 + 3 * 2))

x = 11, y = 16, z = 16

(c) y >>= ((x & 0x2) && z)

x = 10, y = 2, z = 1

(a) 目前使用的格里高利历闰年的规则如下：

~~1. 公元年分非4的倍数，为平年。~~

2. 公元年分为4的倍数但非100的倍数，为闰年。 ✓

~~3. 公元年分为100的倍数但非400的倍数，为平年。~~

4. 公元年分为400的倍数为闰年。 ✓

((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)

请用一个表达式判断某一年是否为闰年。

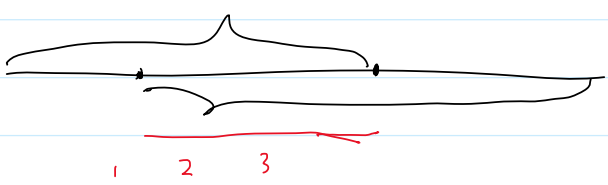
(b) 输入某一天的年月日，输出下一天的年月日。

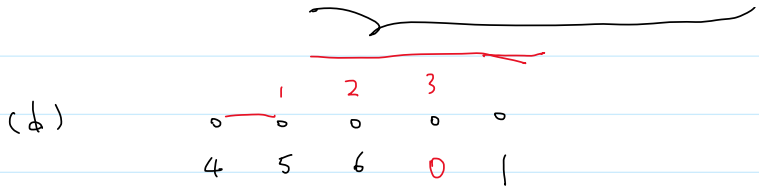
(c) 输入某两天的年月日，输出这两天的相距多少天(不考虑公元前，且第一个日期比第二个日期要早)。

(d) 已知1970年1月1日是星期四，输入之后的某一天的年月日，判断它是星期几?

(b) 类似 bool 操作，

(c)



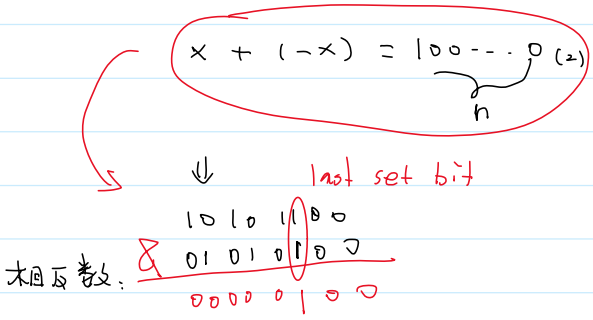


位运算

2024年4月25日 11:02

3. 给定一个值不为0的整数，请找出值为1的最低有效位（last set bit）。

输入：n = 24　0001 1000
输出：8
解释：24的二进制表示为 11000，值为 1 的最低有效位为 2^3。



4. 给定两个不同的整数 a 和 b，请交换它们两个的值（要求不使用中间临时变量）。

(一对逆运算符)

```
printf("a = %d, b = %d\n", a, b);
a = a + b;  // a1 = a0 + b0, b1 = b0
b = a - b;  // a2 = a0 + b0, b2 = a1 - b1 = a0
a = a - b;  // a3 = a2 - b2 = b0, b3 = a0
printf("a = %d, b = %d\n", a, b);
```

+, −　　a + b − b = a

```
printf("a = %d, b = %d\n", a, b);
a = a ^ b;  // a1 = a0 ^ b0, b1 = b0
b = a ^ b;  // a2 = a0 ^ b0, b2 = a1 ^ b1 = a0
a = a ^ b;  // a3 = a2 ^ b2 = b0, b3 = a0
printf("a = %d, b = %d\n", a, b);
```

^, ^　　a ^ b ^ b
= a ^ (b ^ b)
= a ^ 0
= a

5. 给你一个 非空整数数组 nums，除了某个元素只出现一次以外，其余每个元素均出现两次。找出那个只出现了一次的元素。

输入：nums = [1,4,2,1,2]
输出：4

6. 给你一个整数数组 nums，其中恰好有两个元素只出现一次，其余所有元素均出现两次。 找出只出现一次的那两个元素。你可以按任意顺序返回答案（拓展）。

输入：nums = [1,2,1,3,2,5]
输出：[3, 5]
解释：[5, 3] 也是有效的答案。

分类：
a. _, _, ...
b. _, _, ...

(分区) partition

思路：

思路：

xor = a ^ b; （不为0）

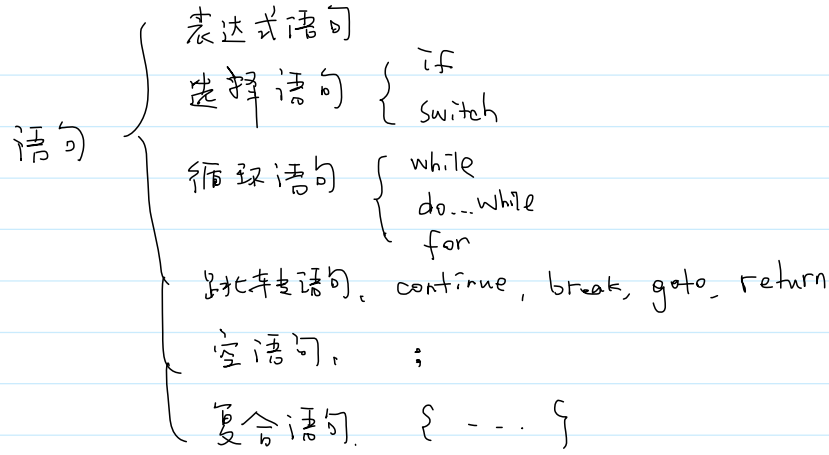
lsb = xor & (-xor); （lsb 是2的幂，a和b在这一位上不同）

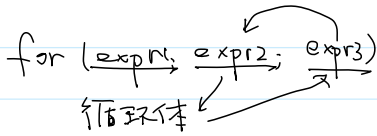
↓

将所有元素分类

语句

2024年4月25日 14:08






```
while ( getchar() != '\n')
    ;
```

switch 语句

2024年4月25日 14:31

数组

2024年4月25日 15:02

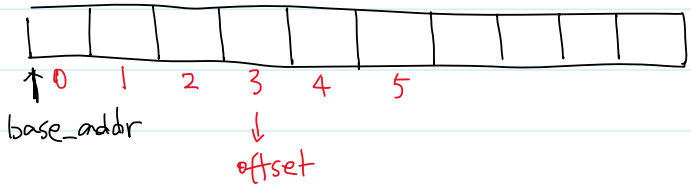
1. 数组的内存模型？

连续的一片内存空间，并且会划分成大小相等的小空间

同一类型

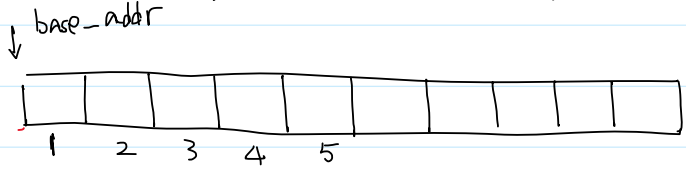
why？

随机访问元素.



i_addr = base_addr + sizeof(elem_type) * i

2. 为什么数组的索引一般是从0开始的？



i_addr = base_addr + sizeof(elem_type) * (i-1);

3. 到极印象：数组效率 > 链表效率？

A. 空间利用率

B. 空间局部性. (数组是连续)

```
int arr[4]; // 声明数组
```

变量名：arr

类型：int [4] → 取值范围 → 内存布局、内存大小

操作

值.

```
int * const p;
```

constant pointer

```
const int * q;
```

pointer to const

```c
int arr1[4] = {1, 2, 3, 4}; // 声明数组
int arr2[] = {1, 2, 3, 4};  // {1, 2, 3, 4}: 数组的初始化式.
int arr3[10] = { 1, 2, 3, 4 };
```

操作: [].

```c
#define SIZE(a) (sizeof(a)/sizeof(a[0]))
```
⇒ 求数组长度.

课堂小练习

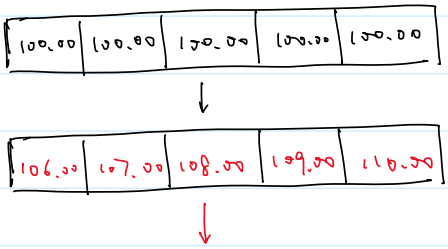
2024年4月25日 16:00

课堂小练习

用户输入初始金额，利率和投资年数，程序将打印一张表格。表格将显示输入的利率以及紧随其后 4 个更高利率下的总金额。程序的会话如下：

```
Enter initial balance: 100
Enter interest rate: 6
Enter number of years: 5

Years     6%     7%     8%     9%    10%
  1     106.00 107.00 108.00 109.00 110.00
  2     112.36 114.49 116.64 118.81 121.00
  3     119.10 122.50 125.97 129.50 133.10
  4     126.25 131.08 136.05 141.16 146.41
  5     133.82 140.26 146.93 153.86 161.05
```

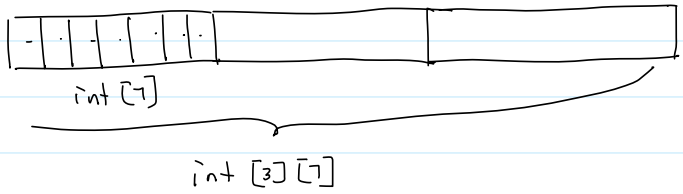


多维数组

2024年4月25日 16:19

结论：C语言只有一维数组。

多维数组本质上也是一维数组。

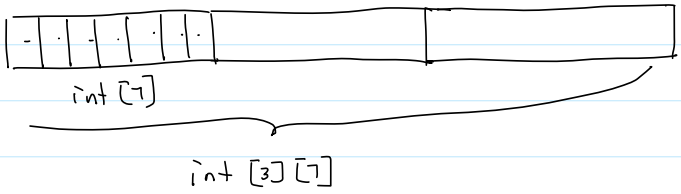



```
int matrix[3][7];
```

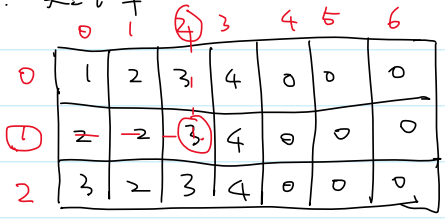
```
int matrix1[3][7] = { {1, 2, 3, 4}, {2, 2, 3, 4}, {3, 2, 3, 4} };
int matrix2[3][7] = { 1, 2, 3, 4, 2, 2, 3, 4, 3, 2, 3, 4 };
int matrix3[3][7] = { 0 };
```

(√)
(×)
(√)

```
int matrix1[3][7] = { {1, 2, 3, 4}, {2, 2, 3, 4}, {3, 2, 3, 4} };
matrix1[1][2];
```



逻辑上：矩阵



常量数组

2024年4月25日 16:36

```
const char suits[4] = { 'S', 'H', 'C', 'D' };
```

Spade：黑桃

Heart：红心

Club：梅花

Diamond：方块

```
// 常量数组：元素不能够修改。
// why：安全(存储静态数组)；效率(编译器可以对常量数组做一些优化)。
```

课堂小练

写一个随机发牌的程序。用户指定发几张牌，程序打印手牌。程序的会话如下：

```
Enter number of cards in hand: 5
Your hand: 9c 7d 3c 5d kd
```

1. 如何表示一张扑克牌？

属性：大小，身高，毛色，上学时间…… （静态）

方法：蹦，跳，撒娇卖萌 （行为）

扑克牌，花色，大小

Suit rank

(2, 7) = 9C 用两个数列表示一张扑克牌。

```
const char suits[4] = { 'S', 'H', 'C', 'D' };
const char ranks[13] = { '2', '3', '4', ..., 'T', 'J', 'Q', 'K', 'A' };
```

2. 怎么随机发牌

和随机数：[0, 3] rand() % 4

[0, 12] rand() % 13

rand

在标头 <stdlib.h> 定义

```
int rand();
```

返回 0 与 RAND_MAX 间的随机整数值（包含 0 与 RAND_MAX）。

srand

在标头 <stdlib.h> 定义

```
void srand( unsigned seed );
```

以值 seed 播种 rand() 所用的随机数生成器。

time

在标头 <time.h> 定义

```
time_t time( time_t *arg );
```

返回编码成 time_t 对象的当前日历时间，并将其存储于 arg 指向的 time_t 对象（除非 arg 为空指针）。

整数 ← time_t

自 1970.1.1 00:00:00 GMT 到现在秒数。

⇓

timestamp 时间戳。

返回编码成 time_t 对象的当前日历时间，并将其存储于 arg 指向的 time_t 对象（除非 arg 为空指针）。

3. 如何避免重复发牌？

bool in_hand[4][13] = {false};

bool in_deck[4][13] = {true}; (×)

函数

2024年4月25日 17:35

数学：返回值、没有副作用

C语言：可以没有返回值，可以有副作用。 function：功能

准则：① 函数的功能应该越单一越好（复用），函数对实现越高效越好。

② 函数是C语言的"基本构建组件"，C语言程序本质就是函数之间调用。

编写程序模拟掷骰子的游戏（两个骰子）。每局游戏的规则如下：

第一次掷的时候，如 果点数之和为 7 或 11 则获胜；如果点数之和为2、3或12则落败；其他情况下的点数之和称为"目标"，游戏继续。在后续的投掷中，如果玩家再次掷出"目标"点数则获胜，掷出7则落败，其他情况都忽略，游戏继续进行。

每局游戏结束时，程序询问用户是否再玩一次，如果用 户输入的回答不是 y 或 Y，程序会显示胜败的次数然后终止。(拓展题，不要求每个同学回答)

```
You rolled: 8
Your point is 8
You rolled: 3
You rolled: 10
You rolled: 8
You win!
```

```
Play again? y
```

```
You rolled: 6
Your point is 6
You rolled: 5
You rolled: 12
You rolled: 3
You rolled: 7
You lose!

Play again? y

You rolled: 11
You win!

Play again? n

Wins: 2 Losses: 1
```