**指针与函数**

1. 函数指针
   1. 用指针变量来替换函数
      1. 代码

int max(int a, int b){

return a>b?a:b;

}

int main()

{

int a = 13, b = 24;

int (\*p)(int, int);

p = max;

int c = p(a, b);

printf("%d", c);

return 0;

}

// 通过地址跟踪，其实我们发现 p 与 max 的地址确是不相等的，为什么呢？？

因为，再很多编译器中，把函数存成一张表格，用表格中的地址来对应函数的真实地址

1. 函数指针做函数参数
   1. 代码
      1. 再来一个更加常用的例子，比如我们用traverse遍历顺序表，然后每遍历一次，把元素打印口来

bool traverse(void(\*func)(int a)){

for(int i = 0; i < list.size(); i++){

func(list[i]);

}

}

void printList(int a){

Printf(“%d”, a);

}

// 调用

traverse(printList);

1. 返回指针值得函数
   1. Int\* func(int, int); 这就是返回指针的函数
   2. 代码

int\* add(int a, int b){

int sum = a + b;

return &sum;

}

int main()

{

int \*result = add(3, 4);

printf("%d", \*result);

return 0;

}

// 但是我们需要注意，我们局部变量，再函数调用结束后，内存是会回收的，也就是说明，sum地址其实已经释放掉了，那么我们就不应该在之前sum的内存中在存值和取值了，因为系统会随时用到那块内存。。最不济就是把sum定义为全局变量