#include<stdio.h>

#include<math.h>

int **main**()

{

    *//pointer 指针，易出错，难懂*

    *//&m是取变量的首地址，但是变量存在不止一个字节的位置，字节数由sizeof等确定*

    *//\*ps来指代s的一级指针，还可以有\*\*pps*

    *//指针变量存储地址，从而可以访问数据。*

    *//指针->与类型有关的地址*

    char c;

    char \*pc=&c;

    *//变量的指针存储首地址，*

    *//指针变量定义：积累型->指针变量访问的数据的类型*

    int \*pi;

    char \*pc1,c,\*pc2;

    void \*p;*//\*是标志，区分指针和普通变量*

    *//&取地址预算，如\*p=&i;*

    *//\*指针运算符，表示获得指针指向的内存数据；*

    int i,\*p=&i;

    (\*p)++;

    *//对于指针的赋值要注意类型匹配*

    *//间接访问*

    int a,b,c,\*pa,\*pb,\*pc;

    pa=&a;

    pb=&b;

    a=100;*//\*pa=100*

    \*pb=200;*//b=200;*

    **scanf**("%d",pc);*//if we input 34, c=34;*

    \*pa=&a;*//将&a放在pa所指的东西里面*

    *//传递地址，指针类型参数的值的传递*

    *//内存数据的操作会改变。//公用内存*

    return 0;

}

Void swap(int \*x,int \*y)

{

 Int t;

 T=\*x,\*x=\*y,\*y=t;

//\*t=\*x,\*x=\*y,\*y=\*t,交换地址，可以将\*x视为一个特殊的容器，其中\*x是一个特别的东西

}

Int main()

{

 Int a=1,b=4;

 Int \*pa,\*pb;

Pa=&a, pb=&b;

 Swap(pa,pb);//即为对于函数中的操作如果是对于指针对应的地址进行操作的话，那么对于原本的地址也会改变。

}

//\*（p+i）对于这个东西的p[i]的调用

允许+，-，++，--；

\*（p++）不是p[1],是先取得p[0]再加一，因此看前置与后置

//p++,p作为当前的操作数，然后后移一个元素，先运算，后增减

//++p,p后移一个元素，然后作为当前的操作数，先增减后运算

//关系运算，如果是p<q,p<=q,p<q:p是否是q的前面，只可以对于同一数据类型的集合来弄；

//p=q,p=q+n,p=q-n

//还包括指针运算，\*&，指向数组的执的下表运算，

/指针与数组

常量指针，不可以a++，

指向数组的指针

Char a[10],\*p;

P=\*a[0];

Char a[10],\*p=&a[0];

Char a[10],\*p=a;

下表运算符[],指针运算符\*

如果q=a+2,那么对于a[i]->\*(q+i-2)

//数组作为实参和形参

F(int x[],int n)//实参，对于int a[100]本身的值有改变

F(int \*x，int n)/形参，对于int a[100]本身的值有改变。

#include<stdio.h>

#include<math.h>

void **sort**(int x[],int n)

{

    int i,j,k,t;

    for(i=0;i<n-1;i++)

    {

        k=i;

        for(j=i+1;j<n;j++)

        {

            if(x[j]>x[k]) k=j;

        }

        if(k!=i)

            t=x[i];x[i]=x[k];x[k]=t;

    }

}

int **main**()

{

    int a[10],\*p,i;

    p=a;//指向数组的指针，便于指向开头的位置

    for(i=0;i<10;i++)

        **scanf**("%d",p+i);

    p=a;

    **sort**(p,10);

    for(p=a,i=0;i<10;i++)

        **printf**("%d",\*p++);

    return 0;

}

指向二维数组的指针

二维数组char a[n][m];

A[i]+j//指向元素的指针；

A+i行指针

Char a[3][4]；

每一行a[0],a[1]a[2],a+i->指向这一行，a+1->对应的行a[1];

\*(a+1)对应的一位数组的首地址；

Char (\*p)[4];指向一维数组的指针；

A[0][3]char

A[1]+3 char \*

A char …[4]

I,j:a[i][j],\*(\*(a+i)+j),\*(a[i]+j)（\*(a+i)）[j]

A是行执政，\*a是指向元素的指针(类比于二重指针）。

11.14

1. 指向数组元素的指针变量 char \*p 根据一维数组和二维数组元素的对应关系，可以访问所有的二维数组元素。

Int a[2][4]->对于a[i][j], 使用int \*a->\*(a+4\*i+j);

指向二维数组元素的指针变量

Char a[m][n];

Char \*p=a[0],&a[0][0],\*a(不可以是a,这个是行指针的第一个东西的地址，我们要

取得是元素，因此要转到元素的指针；注意赋值

P[i\*N+j],\*(p+i\*N+j);

指向二维数组行的指针；

Int a[m][n];

Int (\*p)[N]=a;(此时p对应的是行指针，因此是a;）

P[i]表示数组的第i行，

P[i][j]表示第i行第j列的元素。

1. 一维数组的指针变量， char (\*p)[4],每一行看成一个一维数组

二位数组指针作为函数参数

Void foo(int \*p,int n)；-》

Int a[3][4];

foo(\*a,12);(应用的时候也要取首地址）

Void bar(int (\*p)[4],int n);

Bar(a,3);(行指针只有一种的应勇的方式

指向字符串的指针

与指向数组的指针类似，

1.#include<stdio.h>

#include<math.h>

int main()

{

 char \*s="I love";//==开了一个叫s[]的字符数组来存放i love

 char \*t;

 t="CHina!";

 s[0]='U';

 puts(s);

 s[6]='~';

 puts(s);

 s[12]='~';

 puts(t);

 return 0;

}

Void strcpy（char \*s,char \*t）

{

While(\*t++=\*s++)

}

Int main()

{

Char \*str1=’aaaaa”,str2[20];

Strcpy(str1,str2);

Puts(str2);

}

函数指针

1函数也有相应的内存地址，函数名表示函数的入口地址，

2类型(\*变量名)（【参数类型列表】）

函数名用\*变量名替代，先赋值，后使用，一般用同类型的函数名赋值，不可以算数运算和关系运算。

Int max(int x,int y)

{ return x>y?X:y;}

Int main()

{

Int (\*p)(int, int);//一个指针，指向函数

Int a,b,c;

P=max;

C=(\*p)(a,b);

C=p(a,b);

}

Double intergral(double (\*f)(double),double a,double b)

{\\即为函数内部，对于不同的函数sin,cos,都对应这一个函数，想要求这些东西的积分，这个函数可以完成对于任意的函数的积分求解，即为可以函数内部调用其余的函数。

Double s,h;

Int n=100,i;

H=(b-a)/n;

S=((\*f)(a)+(\*f)(b))/2.0;

For(i=1;i<n;i++;

S+=(\*f)(a+i\*h);

Return s\*h;

}

Y1=intergral(sin,0,0,1,0)

1. 返回指针值的函数;

Int \*foo(int x,int y);=\*（foo(int x,int y)）;

注:练习书写；（找一个题目，写完了敲到机器里面）（每次上机时找一道题目）

复习：周末将两本书看一看，确保两本书看过，（正确性）

Int a[10],\*p;(标志)

\*p=a,&a[0];,a+i,a[i],\*(a+i)

\*p++全是先操作再取地址；

Int a[3][4];

Int \*p;

P=(\*a,&a[0[0,a[0];(元素指针)

P[i\*3+j];找到a[i][j]在一位数组中的偏移量。

行指针 int (\*p)[4];

P=a;

P[i][j];

Int \*foo(int x,int y);

Int (\*foo)(int x,int y);(指向函数的指针；

Int \*f(int \*px,int \*py)

{

 Return \*px>\*py?px:py;

}

\*f(&a,&b)=c;(将c赋给a与b中较大的数

Printf(“%d”,c);

Int \*f(int \*a,int \*b)

{

 Return \*a>\*b?a:b;//比较第一个数组中的数的值。

}

Int I,a[]={1,2,3,4},b[]={5,6,7,8};

For(i=0;i<4;i++

 Printf(“%d\n”,f(a,b)[i])

指针数组和指向指针的指针；

类型 \*数组[length]; char \*p[4]; 而char (\*p)[4]是说一个指针，指向了一个二位数组；

//操作的时候只要对于指针数组操作就可以了/

（修改的时候

Char ste[10][100]={“adsfadsf”,”adsfasdfa”,…};

Char \*name[10];

For(

name[i]=str[i];keyi;

char \*\*p;

p=name;

举例子；

Char \*name[]={“aldksjfah”,”adkljfahkld”…}

Char \*\*p;

Int I;

P=name;

For(i=0;i<3;i++)

 Printf(“%s\n”,\*p++)

Main形式

Int main()

Int main(int argc,char \*argv[]);

Int main(int argc,char \*\*argv)

（）从左往右结合/一个函数

[]从左往右结合，申明一个数组，向外是类型申明；

\*右向左结合，向外是基类型

Void swap1(int \*\*x,int \*\*y)

{

 Int \*p;

 P=\*x,\*x=\*y,\*y=p;

}

Void swap2(int \*\*x,int \*\*y)
{

 Int t;

 T=\*\*x,\*\*x=\*\*y,\*\*y=t;

}

Int main(

{

 Int a=100;b=900;

Int \*p1=&a1,\*p2=&a2;

Swap1(&p1,&p2);

}

11.26

Stu.name=”Mike”,这句话不对！

结构体变量之间可以直接赋值

结构体做了一段就数出来

链表

Head->a1 \*->a2 \*->….->a2 NULL

链式，boot IOS app,文件的链式存储

Struct node

{

 Int data;

Struct node \*next;

};

Struct node \*head;

Malloc,calloc,free函数

#include<malloc.h>,<stdlib.h

Malloc<size>:size是分配存储空间的字节

若成功，返回指向起始地址的指针，反之返回NULL

Calloc(n,size):

N:分配内存的项目书

Size:分配每个东西的字节数

Free(ptr)ptr:要释放的内存区地址，是mallc,calloc的返回值，

动态数组

链表

制表

空表，只有一个节点，对于第一个节点操作，对于最后一个节点操作。

最后一个节点的next执政是NULL

12.3.2024文件

File 对于文件的存取是以字节为单位（1）文本文件 ASCLL吗存储 （2）二进制文件

FILE \*fp;（文件类型指针）

FILE \*fopen(char \*filename,char \*mode);

Filename：要打开的文件路径，mode（打开模式），若打开成功，则返回指向被打开的文件的指针，反之返回NULL(0)

D：\wang\...自己建一个文件夹

Current内部指针，一般不用操作，而这里是fopen的返回值。

R只读，w只写，创建or打开，覆盖原本的文件，a追加，在文件的末尾追加。

R+,读写，打开已有的文件，w+读写，创建or打开，覆盖原本的文件，a+读写，创建or打开，在末尾追加，t，b,

用了fopen就要fclose

Int fclose(\*fp),fp是前面打开的文件指针。

成功返回0，错误返回-1；

FILE \*fp1,\*fp2,\*fp3;

 char filename[]="file3.dat";

 if((fp1=fopen("aaa.txt","r")))

 {

 printf("Cannot Open This File!\n");

 exit(0);

 }

fclose(fp1);

feof(\*fp)(判断一个文件是否结束了，若结束，返回非零值，否则返回0)

fput(int c,FILE \*fp);(c为读入的东西，fp是文件指针，若失败返回EOF)

fwrite,fread(void \*buffer,size\_t size, size\_t count, FILE \*fp);

buffer：读写的数据快的地址，size：字节数，count：数据项个数

#include<stdio.h>

#include<math.h>

#include<string.h>

#include<ctype.h>

#include<stdlib.h>

#include<vector>

using namespace std;

int main()

{

 FILE \*fp;

 int i;

 struct std s[NUM];

 fp=fopen("students.dat","rb+");

 for(i=0;i<NUM;i++)

 fwrite(s+i,sizeof(struct std),1,fp);

 for(i=0;i<NUM;i++)

 fread(s+i,sizeof(struct stud),1,fp);

 printf("%d",s[i].111);

 fclose(fp);

 return 0;

}

少量的信息 fscang,fprintf

Fscanf(

箭头是优先级最高的

**考试要求**

**选择题，单选 9，多选 6(不用花太多时间)**

**阅读程序写结果，15分，没题3分(一定要拿到)**

**程序天空 15分，每个题目1/1.5分，可能有几个小题比较难，大部分比较简单**

**算法描述 8分 （流程图，伪代码**

**编程题 47分，4题，7 8 12 20（写几个子函数)**

**字符串， 函数(指针函数，函数参数传递，形参，调用时指针)**

**链表(大题), 大题(写文件，将一个结果写到一个文件里面**

**数据量小的时候用txt,fprintf,fscanf**

**结构题数组 fwrite(编程题留出1h 10min)**

**题目**

**Createlist()键盘输入。。。并用头插法，当学号0时，每输入一个时都提示（若有重复，就推出）。10**

**删除节点，找到后删除，no found， （对于找不到的时候要报出信息）**

**12.12.2024**

静态局部变量，全局在外面

Int f(int a)

{

 Int b=0;

 Static int c=3;

 B++;c++;//**静态局部变量存在静态存储区，不会回收，值会保留。**

 Return a+b+c;

}

Void main()

{

 Int x=2,I;

 For(i=0;i<3;i++

 Printf(f(x))

}

I 0 x 2 a 2 b 1 c 4 f 7

I 1 x 2 a 2 b 1 c 5 f 8

I 2 x 2 a 2 b 1 c 6 f 9

void func(int \*x,int \*y)

{

 \*x+=\*y;

 \*y=\*x-\*y;

 \*x-=\*y;

}（交换）

int sub(int m,int n)

{

 int z;

 if(n==0) z=m;

 else z=sub(n,m%n);

 return z;

}（gcd）

int main()

{

 int a[]={19,11,30,66,75,95};

 int \*ptr=a;

 for(;;ptr++)

 {

 if((ptr-a)==sizeof(a)/sizeof(int))

 {

 break;//if end break;

 }

 if(\*ptr%5==0) continue;

 printf(\*ptr)//19 11 66

 }

 return 0;

}

int main()

{

 static struct t{char c[16],\*t;}t1={"China","Asia"};

 static struct tt

 {

 char \*cp;

 struct t cc;

 }t2={"Beijing",{"shanghai","Hefei"}};

 printf("%c%c\n",t1.c[0],\*t1.t);//CA

 printf("%s %s\n",t1.c,t1.t);//China Asia

 printf("%s %s %s\n",t2.cp,t2.cc,t2.cc.t);//Beijing shanghai

 printf("%s %s\n",++t2.cp,++t2.cc.t);

 return 0;

}

T2.cc-> Shanghai,输出的东西是第一个东西。

int main()

{

 int i=10;

 switch(i)

 {

 case 9:i+=1;

 case 10: i+=1;11

 case 11: i+=1;12

 default: i+=1;13

 }

 //answer=13;注意对于switch中有没有break；

 return 0;

}

struct{int x,y;}s[2]={{1,2},{3,4}};,\*p=s;

 ++p->x 2;

 (++p)->x 3;

 struct {int x;char \*y;}tab[2]={{1,ab,2,cd

 };

 }; \*p=tab;

 \*p->y a;

 \*(++p)->y c;

Int a=10,c=9;

(--a==c++)?—a:++c 值为 8, a 8 c 10

Int x=3,y=2,z=1;

X=y++>=x&&x-y==++z;

X=0,y=3,z=1;

先进行比较运算符，后进行赋值操作；

Z=0+1+2=3；x=0;y=y\*3=6;

X0 y6 z3;

X=y=z=1;

--x&&++y（0）||z++; 0 1 2；

Int i=4,j=4,k=8;

\*p=&I,\*q=&j,\*r=&k;

Int x,y,z;

X=p==&i(x=1);

Y=3\*-\*p/\*q+7;y=3\*(-3)/4+7=-2+7=5;

Z=\*(r=&k)=\*p\*\*q;//24;

int digit;

 long in,s;

 scanf("%ld",&in);

 s=0;

 if(in<0) in\*=-1;//注意最后情况是负数的情况。

 while(in>0)

 {

 digit=in%10;

 s=s+digit\*digit;

 in=in/10;

 }

 printf("")

return 0;

输入一个英文自负床，统计出a-z出现次数，并写入正文文件中。

int xtoi(char s[])

{

 int k,d=0;

 for(k=0;s[k]==' '||s[k]=='\t';k++)

 ;

 while(s[k]>='0'&& s[k]<='9' || s[k]>='a' && s[k]<='f')

 {

 if(s[k]>='0' && s[k]<='9')

 d=d\*16+s[k]-'0';

 else

 d=d\*16+s[k]-'a'+10;

 k++;

 }

 return d;

}