学习进度汇报

一、C语言知识点

1、常量

- 整型常量：正数、负数、0
- 实型常量： 所有带小数点的数字
- 字符常量： 单引号引起来的字母、数字、英文符号
- 字符串常量： 双引号引起来的

格式控制符	说明
整型	%d
实型	%f
字符	%c
字符串	%s

2、变量

- 先定义再赋值
- 只能存一个值
- 变量名不允许重复定义
- 一条语句可定义多个变量 -

	数据类型符
整数	short、int、long、long long
小数（实数、浮点数）	float、double、long double
字符	char

3、标识符

- 由数字、字母、下划线（—）组成
- 不能以数字开头
- 不能是关键字

4、运算符

- 整数运算，结果一定是整数，想得到小数必须有小数参与运算
- 取余：运算的数据必须全部是整数
- 隐式转换: 将取值范围小的转成取值范围大的
  double>float>long long>int>short>char

- 强制转换:将取值范围大的转成取值范围小的
  格式:目标数据类型 变量名=（目标数据类型）被强转的数据

```
int b=10;
short i=(short)b;
```

- 运算符

运算符	符号
自增自减运算符	++、--
赋值运算符	=、+=、-=、*=、/=、%=
关系运算符	==、!=、>、>=、<、<=
逻辑运算符	&&、！
逗号运算符（分隔符）	,

- 优先级：小括号优先
  逗号运算符,从左到右，优先级最低
  一元>二>三元
  &&>||>赋值

5、流程控制语句

顺序结构：从上往下依次执行
分支结构：
if语句：

```
if(关系表达式)
  {
   语句体;
  }

  或
  if(关系表达式a)
  {
      语句体a;
  }
  else if(关系表达式b)
  {
      语句体b;
  }
  ---
  else
  {
      语句体c;
  }
```

switch语句：

```
        switch(表达式)
            {
                case  值1:     //值只能是字面量，不能是变量
                  语句体1;
                  break;      //表示中断，结束


                case  值2:
                  语句体2;
                  break;

                ---
                default:     //所有情况都不匹配时执行
                   语句体n;
                   break;

               }
```

if与switch:

- switch:有限个case匹配
- if:一般对一个范围进行判断

循环结构：

```
        for循环：
            for(初始化语句;条件判断语句;条件控制语句)
            {
                循环体语句
            }

      例：for(int i=1;i<=100;i++)
          {
            printf("123");
          }


         while循环：
             while(条件判断语句)
             {
                循环体语句
                条件控制语句
             }
```

for与while的区别：

- for循环中知道循环次数或范围
- while循环中不知道循环次数和范围，只知道结束条件

无限循环：

```
for( ; ; )
{
    printf(" ");
}


while(0)     //o表示不成立，1表示成立
{
    printf(" ");
}
```

跳转控制语句：在循环过程中，跳到其他语句上执行

- break：不能单独书写，只能写在switch，或循环中，表示结束、跳出
- continue：结束本次循环，继续下次循环，只能写在循环中

6、函数

函数:程序中独立的功能
基本定义格式：

```
返回值类型 函数名(形参1,形参2,...)      //若无返回值可使用 void
{
 函数体;
 return 返回值;          //return作用：1.结束函数 2.把后面的数据交给调用处
}
```

7、随机数

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int main()
{
    srand(time(NULL));    //使用srand设置种子
    int num=rand();       //使用rand获取随机数
    printf("%d",num);
    return 0;
}
```

8、数组

- 数组：一种容器，可以用来储存同种数据类型的多个值
    - 特点1：连续的空间
    - 特点2：一旦定义，长度不可变
- 数据类型 数组名 [长度] ={数据值,...}
    - 长度省略：数据值的个数就是数组长度
    - 长度未省略：数据值的个数<=数组长度,空位默认值为0（若字符串中则为NULL）
- 元素访问
    - 索引：数组的一个编号，从0开始，连续+1
    - 获取：数组名[索引] arr[s]
    - 修改：数组名[索引]=数据值； arr[s]=10;
    - 数组遍历：

```
for(int i=0;i<=4;i++)
{
   printf(" ");
}
```

9、内存和内存地址

- 内存：软件在运行时，用来临时储存数据的
- 内存地址：内存在每一个小格子的编号（书写时转成16进制），可快速管理内存空间
- 获取变量中的内存地址：

```
int a=10;
printf("%p\n",&a);
```

- 数组的内存地址：是第一个元素的第一个字节空间的地址，也是首地址
- 索引：偏移量
- 通过变量的首地址，不能确定变量中储存的数据
- 首地址只能获得第一个字节的数据，不能获得完整的，要通过首地址+数据类型才能知道占用多少字节

10、数组常见算法

- 基本查找/顺序查找
- 二分查找/折半查找
    - 前提条件：数组中的数据必须是有序的
    - 核心逻辑：每次会排除一半的查找范围

- 插值查找
  要求：数据要有序，且数据分布尽可能均匀 优势：满足要求，效率比二分查找高
- 冒泡排序
  相邻的数据两两比较，小的放前面，大的放后面
- 选择排序
  从0开始，拿着每一个索引上的元素与后面的元素依次比较

11、指针

- 指针=内存地址
  定义格式：数据类型*变量名
  例：

```
int a=10;
int\*p=&a;
```

- 查询数据、储存数据

```
    int a=10;
    int*p=&a;
    printf("%d\n",*p);  //查询数据
    *p=200;   //储存数据、修改数据
    //*p为解引用运算符（通过后面的内存地址去获取对应的数据）
```

- 注意
  - int*p中名字是p不是*p,*只是标记，标记p里存的是内存地址
  - 指针变量的数据类型要跟指向变量的类型保持一致
  - 指针变量占用的大小与数据类型无关，与编译器有关
  - 给指针变量赋值时，不能把一个数值赋值给指针变量
- 指针的作用：
  1、操作其他函数中的变量（调用函数时仅把变量的值传递过去，在函数中改变了值，而对原变量无影响）
  2、函数返回多个值
  3、将函数的结果和状态分开

12、指针高级

- 步长:指针移动一次的字节数
- 指针运算：

```
加法：指针向后移动N步
减法：指针向前移动N步
```

```
有意义操作：
      指针与整数进行加减操作
      指针与指针进行减操作（间隔步长）
无意义操作：
      指针与整数进行乘除操作（此时指针指向不明）
      指针与指针进行加乘除操作
```

- 数组指针：

```
概念：指向数组的指针
作用：方便地操作数组中的各种数据
例：int(*p)[5]=&arr
```

- 指针数组：

```
概念：存放指针的数组
作用：用来存放指针
例：int*p[5]
```

- 获取数组的指针：

```
int arr[]={};
int*p1=arr;

for(int i=0;i<len;i++)
{
  printf("%d\n",*p1++)
}
```

```
arr参与计算时，会退化为第一个元素的指针
特殊情况：sizeof运算时不会退化，arr还是整体
        &arr获取地址时，不会退化
        步长=数据类型*数组长度
```

- 二维数组：把多个小数组，放到一个大的数组中

```
定义格式：
      数据类型 arr[m][n]=
      {
       {1,2,3}
       {1,2,3}
```

```
 {1,2,3}
}     //m：二维数组长度   n：一维数组长度
```

- 索引遍历

```
方法一：
    int arr[3][5]=
    {
      {1,2,3,4,5}
      {11,22,33,44,55}
      {111,222,333,444,555}
    };

    for(int i=0;i<3;i++)
    {
      for(int j=0;j<5;j++)
      {
        printf("%d",arr[i][j]);
      }
    }
方法二：
    int arr1[3]={...};
    int arr2[5]={...};
    int arr3[9]={...};
    int*arr[3]={arr1,arr2,arr3};     //储存的是内存地址

    int len1=3;
    int len2=5;
    int len3=9;
    int lenarr[3]={len1,len2,len3};

    for(int i=0;i<3;i++)
    {
      for(int j=0;j<lenarr[i];j++)
      {
        printf("%d",arr[i][j]);
      }
    }
```

- 指针遍历
  数组指针的数据类型，要跟数组内部元素保持一致
  例： int(*p)[5]=arr

```
   方法一：
    for(int i=o;i<3;i++)
    {
     for(int i=0;i<5;j++)
     {
       printf("%d",*(*p+j));
```

```
        }
        printf("\n");
        p++;
       }
```

```
方法二:
 int arr1[5]={1,2,3,4,5};
 int arr2[5]={6,7,8,9,0};
 int*arr[2]={arr1,arr2};
 int* *p=arr;    //数据类型*指针名=arr

 for(int i=o;i<2;i++)
 {
  for(int i=0;i<5;j++)
  {
    printf("%d",*(*p+j));
  }
  printf("\n");
  p++;
 }
```

二、易错点

1、语句结束时要加“；”
2、int main()后要加{}
3、使用scanf，变量前要加&
如：scanf("%d",&a);
4、long long类型对应占位符为%lld
5、定义函数时，形参用“，”隔开 定义时不用加“；”，调用时不加int，要加“；”
6、C语言小数默认为double类型，如果赋值float类型，要加后缀f或F
7、unsigned只能与整数类型结合
8、键盘录入时，scanf后不能加\n,占位符和后面的变量要一一对应（包括中间的所有符号）
9、整数运算，结果一定是整数，想得到小数必须有小数参与运算
有参与小数计算，结果一定是小数
10、取余时，运算的数据必须全部是整数，获取余数的正负与被除数保持一致
11、\n:回车换行符，光标会移到下一行行首
\r:回车符，光标会移到这一行行首
12、在使用循环时，要注意变量的生命周期、变量的值是否改变
13、同一变量不能重复定义，如果修改可g=g+h
14、case穿透只会往下穿，不会穿到上面 如果遇到break，会直接结束整个switch
15、变量的生命周期：变量只在所属的大括号中有效

三、遇到问题解决方法

1、再次回顾所学内容，查询是否有遗漏、遗忘
2、多尝试，试试不同组合、不同方式，并尝试理解

3、使用各种方式查询
4、问Deepseek
5、记录下每次问题及尝试，并进行对比，思考问题出现的原因