day4: 结构体和共用体

【1】
struct 结构体类型名
{
数据类型　　成员名1；
数据类型　　成员名2；
：
数据类型　　成员名n；
}；

【2】定义结构体变量

struct 结构体名
{
　　成员列表；
}变量名1；

struct 结构体名 变量名2；

【3】访问结构体成员
结构体变量名.成员名

【4】结构体初始化操作

struct 结构体名 变量名={初始数据表}；

【5】无名结构体
没有类型名。
一般状态下：无名结构体不可以定义局部变量。

【6】结构体嵌套
1--- 成员所属的数据类型是 结构体类型
2--- 内部的结构体通常定义为无名结构体

【7】结构体数组
本质是数组，元素是结构体类型

【8】结构体指针

本质是指针，指向一个结构体数据

【9】typedef

用于给已存在的类型名，进行重命名。

【10】共用体
1--- 定义

union 共用体类型名
{
成员表列；
}变量名1；
union 共用体类型名 变量名2；

2--- 成员共用同一个存储区域；

数据对齐：
一般放在偶地址

【11】
生命期： 变量占据内存的时间期限－－－－静态变量和动态变量（函数、程序）
作用域：变量出现的有效区域－－－－－局部变量和全局变量

关键字	变量位置	存储期	作用域	存储模型
1、auto(默认)	局部变量	函数同步	函数内	自动存储
2、register	局部变量	函数同步	函数内	寄存器存储
3、static	局部变量	程序同步	函数内	空链接 1、程序同步 2、初始化一次 3、默认初始化为0
	全局变量	程序同步	本文件内	内部连接
4、extern	全局变量	程序同步	多文件	外部链接

【12】gcc

1--- gcc hello.c -o hello

2--- 预处理、编译、汇编、链接
gcc -E hello.c -o hello.i
gcc -S hello.i -o hello.s
gcc -c hello.s -o hello.o
gcc   hello.o -o hello
3--- 汇编、链接
gcc -c hello.c -o hello.o
gcc   hello.o -o hello

【13】 malloc（手动分配和释放）

void *malloc(size_t size);
    功能： 申请 size个字节大小的空间
    返值： 就是这段申请空间的首地址

    void free(void *ptr);
    功能：释放堆区手动申请的空间

malloc 和 free 一般是配对使用。有申请一般都有释放。

【14】 Makefile

target ： dependency_files
<TAB> command

1---
gcc -c main.c -o main.o
gcc   main.o -o main

main:main.o
main.o:main.c

2---
gcc -c main.c -o main.o
gcc -c add.c -o add.o
gcc -c sub.c -o sub.o
gcc *.o -o main

$< 第一个依赖文件的名称
$+ 所有的依赖文件，以空格分开，并以出现的先后为序
$^ 所有不重复的依赖文件，以空格分开
$@ 目标文件的完整名称
$* 不包含扩展名的目标文件名称
$? 所有时间戳比目标文件晚的的依赖文件,并以空格分开

[14] others

初始化的全局变量，局部变量以及为初始化的全局变量、静态变量存在什么段

初始化为0又是什么状况

http://blog.csdn.net/canbus/article/details/8660065

bss段、data段和text段

day4: homework
主函数里，创建一个局部变量的结构体数组，从终端依次获取
子函数实现一个功能： 按照学生的成绩递增进行排序