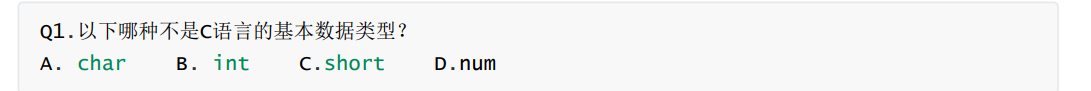
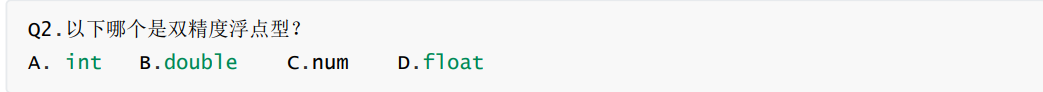
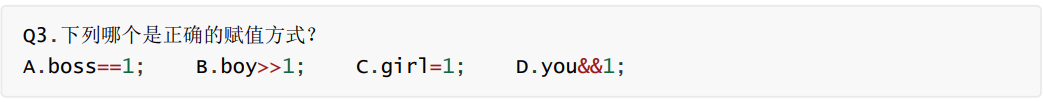
1. D



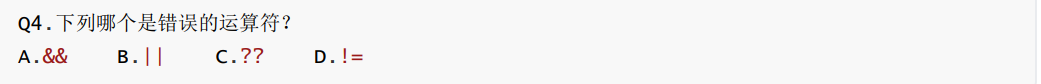
1. B



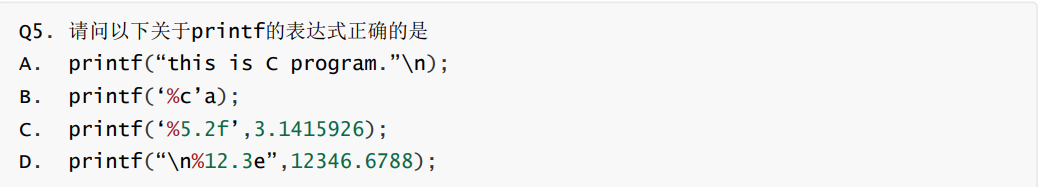
1. C



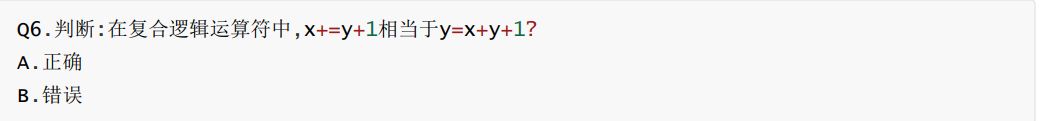
1. C



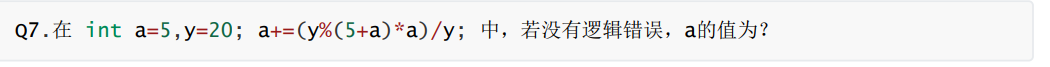
1. D



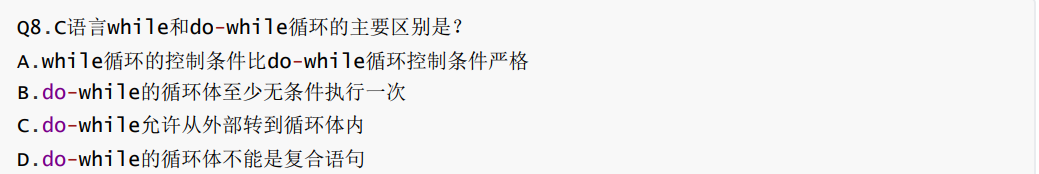
1. B



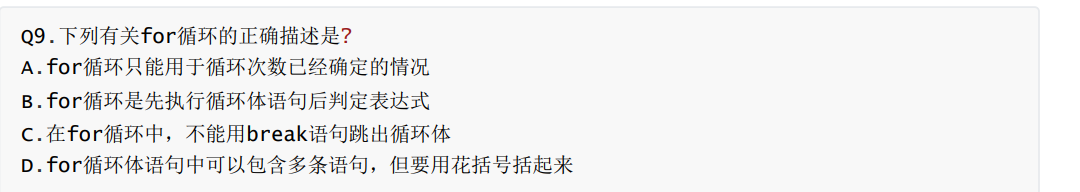
1. a=5



1. B

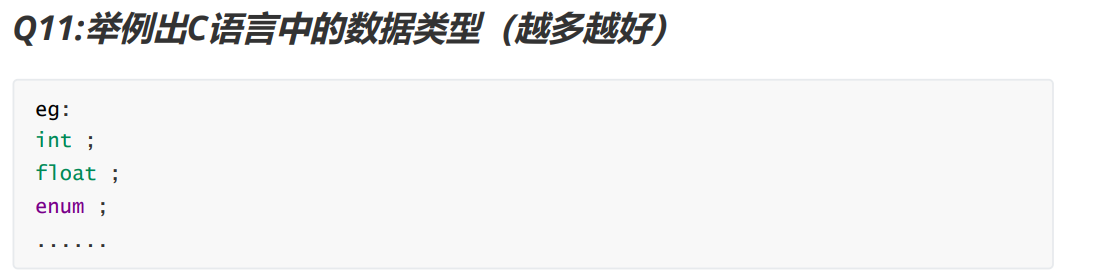


1. D





1. break是跳出一个循环体或完全结束本次循环，不执行后面的循环；continue是跳过这次循环，不执行本次循环中未执行的语句，执行下一次循环。



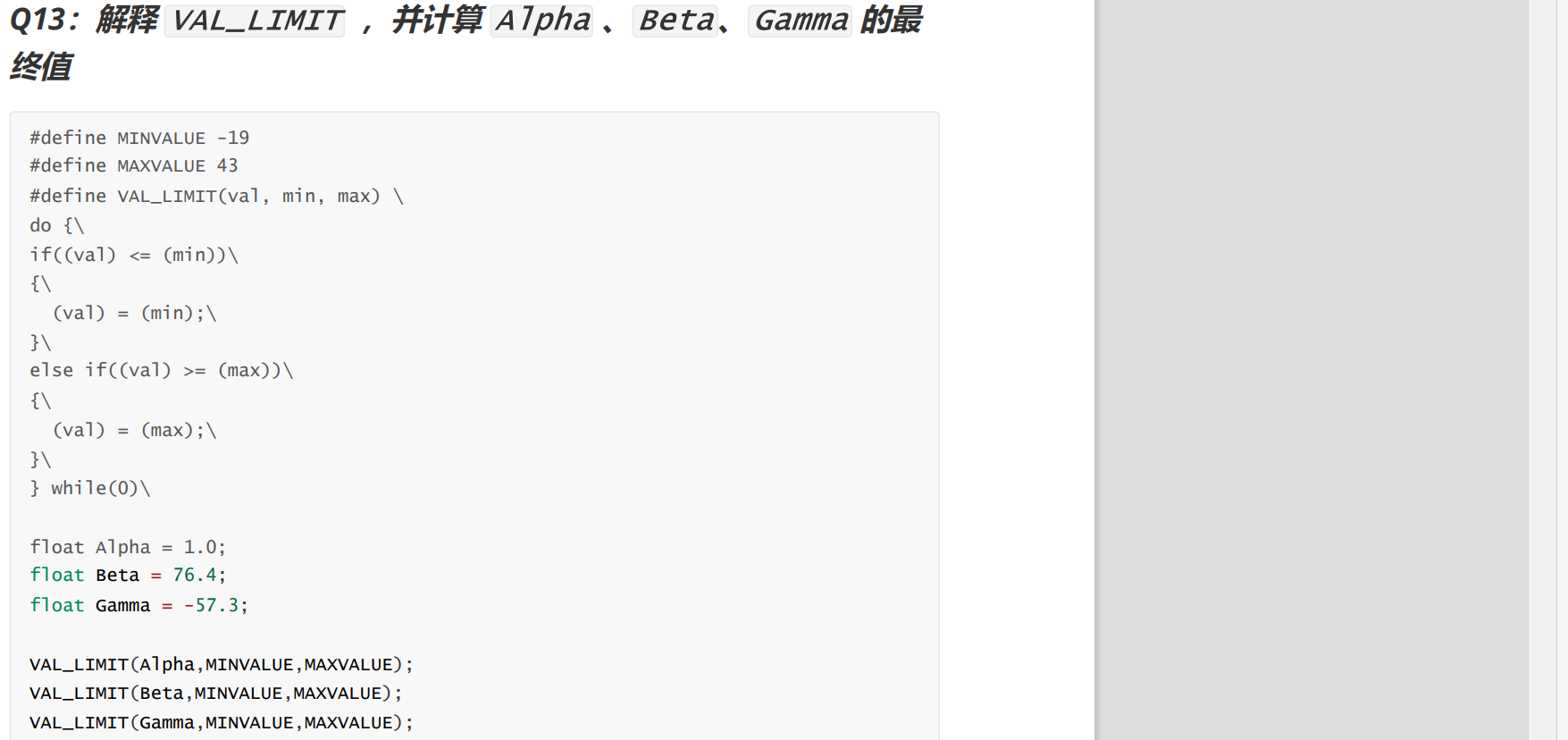
1. short、int、long int、long long、float、double、char、enum、union、struct



1. while(1)

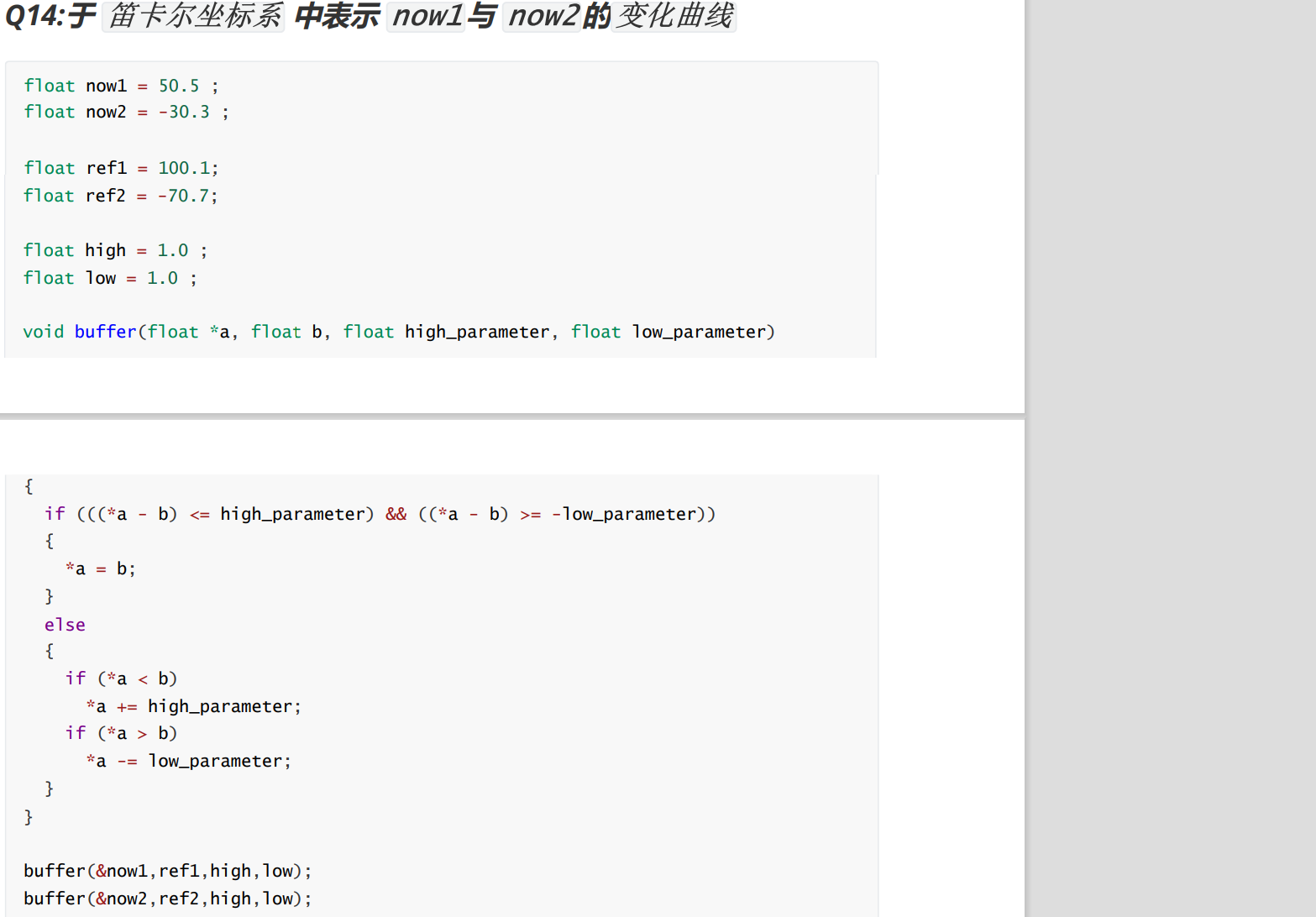
{

}



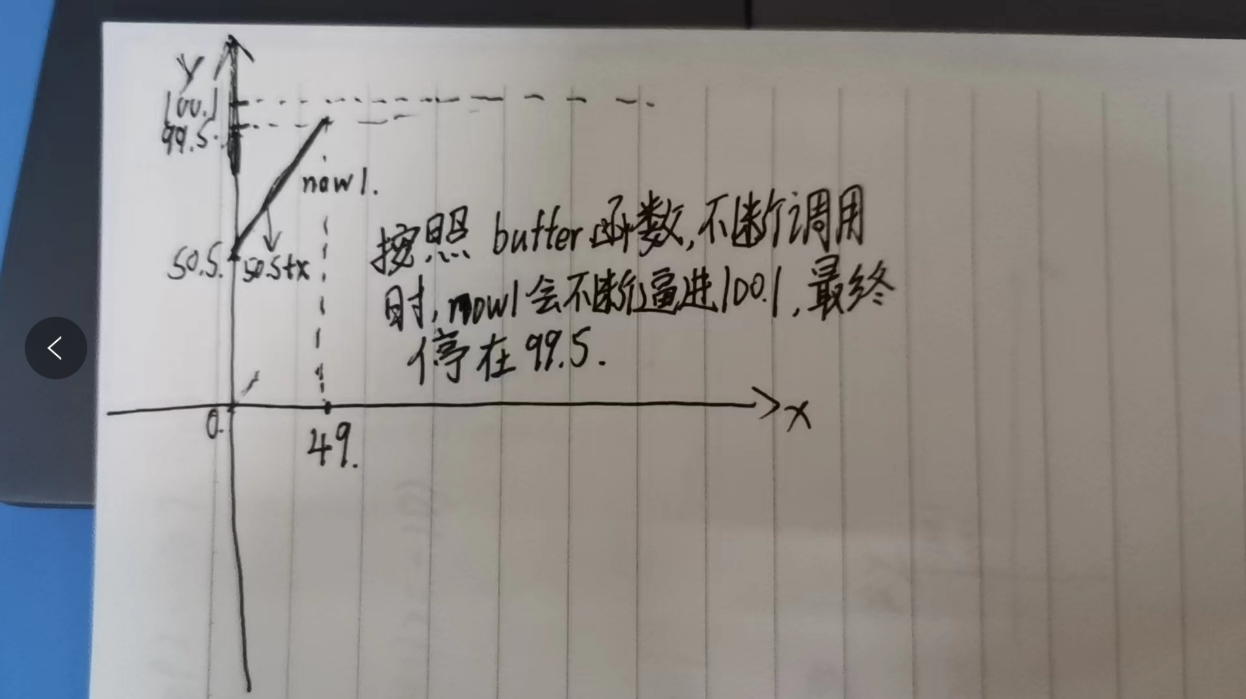
1. VAL\_LIMIT（val，min，max）被定义成一个do{…}while（0）结构，作用是检查val的值是否在min和max之间（包括min和max）。如果val小于min，则将其设置为min；如果val大于max，则将其设置为max。

Alpha最终值为1；Beta最终值为43.0,；Gamma最终值为-19.0

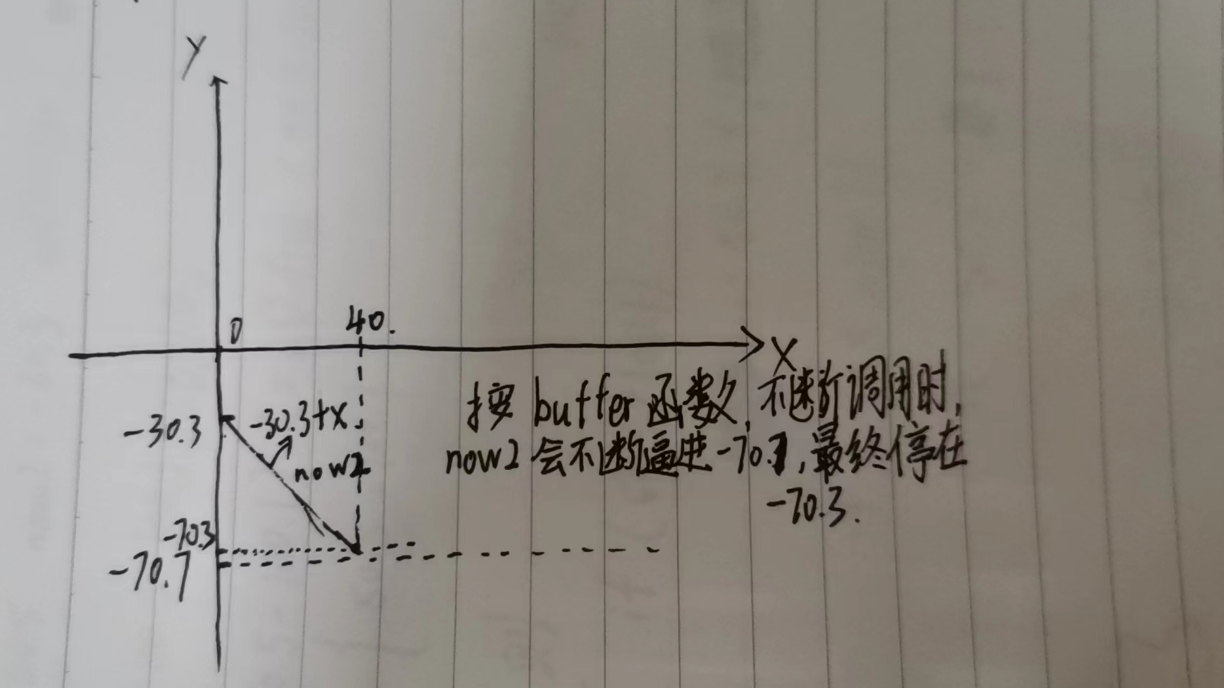


14.

now1



now2



15.

#include<stdio.h>

typedef unsigned int uint32\_t;

typedef struct

{

uint32\_t get\_count;

uint32\_t set\_count;

float out;

}ramp\_t;

//初始化

void ramp\_init(ramp\_t \*ramp,uint32\_t target\_count)

{

ramp->get\_count = 0;

ramp->set\_count = target\_count;

ramp->out = 0;

}

//斜坡式上升

float ramp\_calc(ramp\_t \*ramp)

{

if(ramp->set\_count <= 0)

return 0;

if(ramp->get\_count >= ramp->set\_count)

ramp->get\_count = ramp->set\_count;

else

ramp->get\_count++;

ramp->out = (float)ramp->get\_count/(float)ramp->set\_count;

return ramp->out;

}

int main()

{

ramp\_t temp;

//定义自变量x

float x=0;

//初始化

ramp\_init(&temp,10);

int i=0;

printf("斜坡式增长到1：\n");

for(i=0;i<10;i++)

{

x=ramp\_calc(&temp);

printf("%.2f\n",x);

}

//初始化

ramp\_init(&temp,10);

printf("斜坡式下降到1：\n");

for(i=0;i<10;i++)

{

x=-ramp\_calc(&temp);

printf("%.2f\n",x);

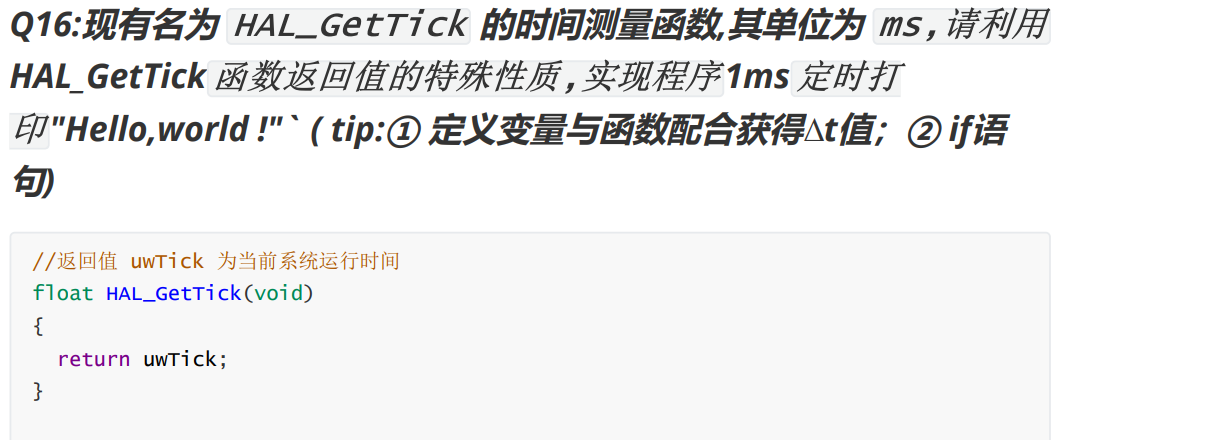
}

return 0;

}

运行示意图：





16.

//返回值 uwTick 为当前系统运行时间

float HAL\_GetTick(void)

{

return uwTick;

}

int main()

{

//初始化上次系统时间

float Last\_time=HAL\_GetTick();

while(1)

{

//记录当前系统时间

float Current\_time=HAL\_GetTick();

//时间差

float differ=Current\_time-Last\_time;

if(differ>=1)

{

printf("Hello,world\n");

//更新时间

Last\_time=Current\_time;

}

}

return 0;

}



17.

#include<stdio.h>

#define VALUE\_B 217

#define VALUE\_C 219.321

static int alpha;

static float beta;

void STDRxCallback(int\* b, float\* c)

{

\*b = VALUE\_B;

\*c = VALUE\_C;

}

void (\*callback)(int\* c, float\* b);

int main()

{

//赋值

callback=STDRxCallback;

//初始化

callback(&alpha,&beta);

printf("alpha:%d\n",alpha);

printf("beta:%.3f\n",beta);

return 0;

}



18.

#include<stdio.h>

typedef unsigned char uint8\_t;

typedef unsigned short int uint16\_t;

uint8\_t alpha = 0;

uint8\_t beta = 0;

uint16\_t gama = 20001;

int main()

{

//分离高八位

beta=gama >>8;

//分离低八位

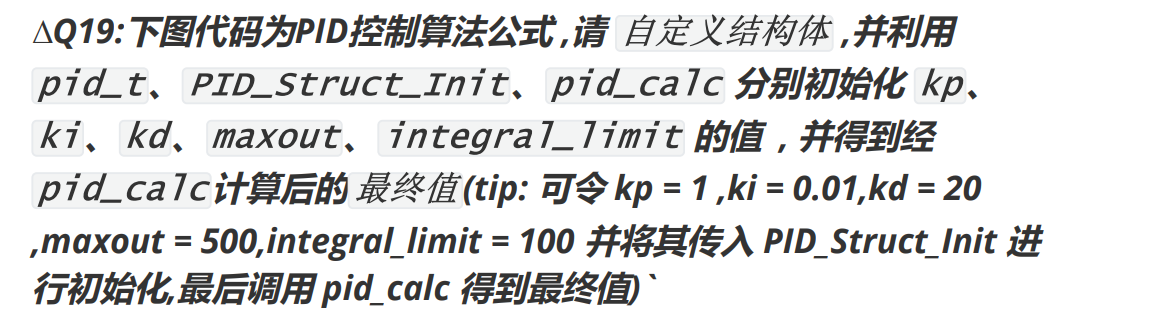
alpha=gama & 0xFF;

printf("高八位：%d\n",beta);

printf("低八位：%d\n",alpha);

return 0;

}



19.

#include<stdio.h>

typedef int int32\_t;

//

题目给的函数

//

int main()

{

float final=0;

pid\_t pid;

//初始化

PID\_Struct\_Init(&pid,1,0.01,20,500,100);

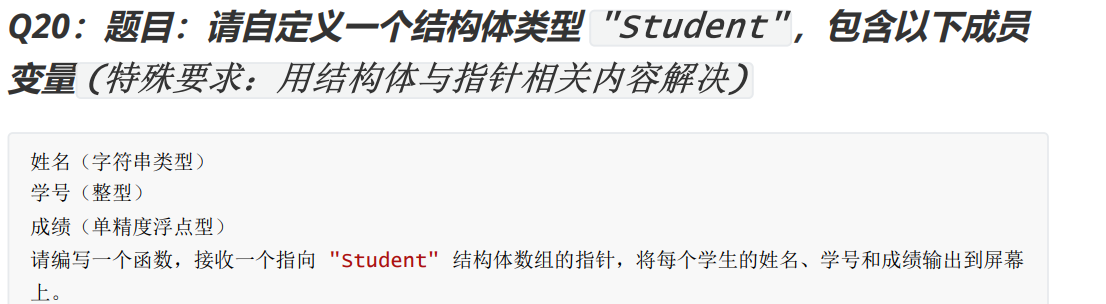
//计算

final=pid\_calc(&pid,10,50);

printf("%f",final);

return 0;

}



20.

#include<stdio.h>

typedef struct S

{

char name[20]; //姓名

int num; //学号

float score; //成绩

}Student;

//输出函数

void Print(Student \*stu)

{

printf("姓名 学号 成绩\n");

int i=0;

for(i=0;i<2;i++)

{

printf("%s %d %.2f\n",stu[i].name,stu[i].num,stu[i].score);

}

}

int main()

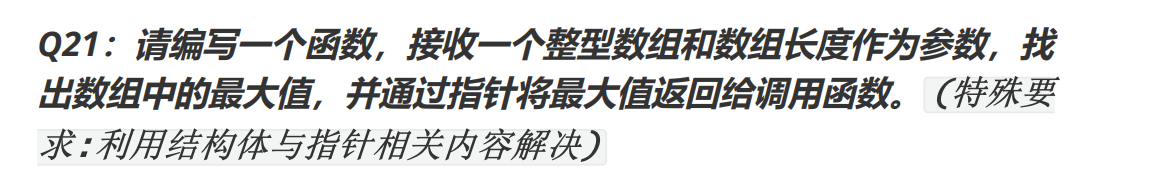
{

Student stu[2]={{"小红",101,94.5},{"小明",102,89.3}};

Print(stu);

return 0;

}



21.

#include <stdio.h>

void FindMax(int \*arr,int length,int \*max);

int main()

{

int arr[] = {8, 11, 4, 2, 6};

int length = sizeof(arr) / sizeof(arr[0]);

int max;

//在此填写函数 ↓

FindMax(arr,length,&max);

//在此填写函数 ↑

printf("最大值：%d\n", max);

return 0;

}

void FindMax(int \*arr,int length,int \*max)

{

int i;

\*max=arr[0];

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

{

if(arr[i+1]>\*max)

\*max=arr[i+1];

}

}



22.

#include <stdio.h>

//> 注意：不准在这给变量赋值，只可给函数声明！

void Control(int \*nowspeed,int maxspeed);

int main() {

int nowspeed=5000,maxspeed=4000;

//请插入一个void函数 ↓

Control(&nowspeed,maxspeed);

//请在此插入一个函数 ↑

printf("nowspeed:%d\n",nowspeed);

return 0;

}

//请在此补充函数↓

//第一个参数为电机当前转速，第二个参数是最大限速

void Control(int \*nowspeed,int maxspeed)

{

if(\*nowspeed>maxspeed)

{

\*nowspeed=maxspeed;

}

else if(\*nowspeed<-maxspeed)

{

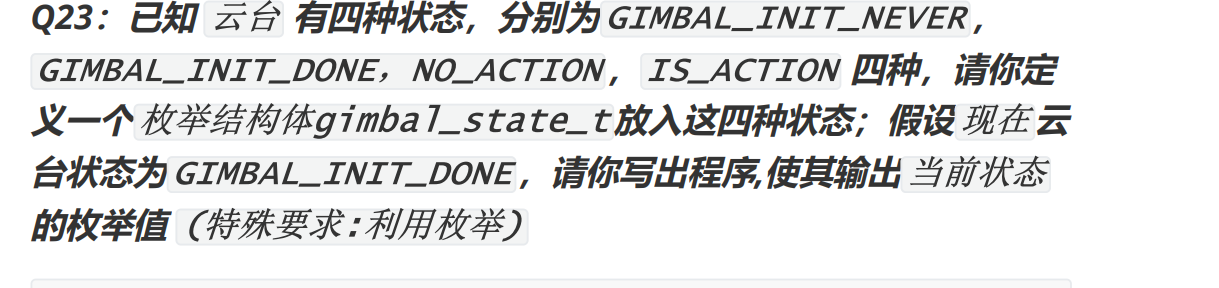
\*nowspeed=-maxspeed;

}

}

//请在此补充函数↑

//> 注意:①不可加入全局变量!



23.

#include <stdio.h>

typedef enum

{

GIMBAL\_INIT\_NEVER=0,

GIMBAL\_INIT\_DONE,

NO\_ACTION,

IS\_ACTION

}gimbal\_state\_t;

int main()

{

//定义枚举变量

gimbal\_state\_t gimbalState;

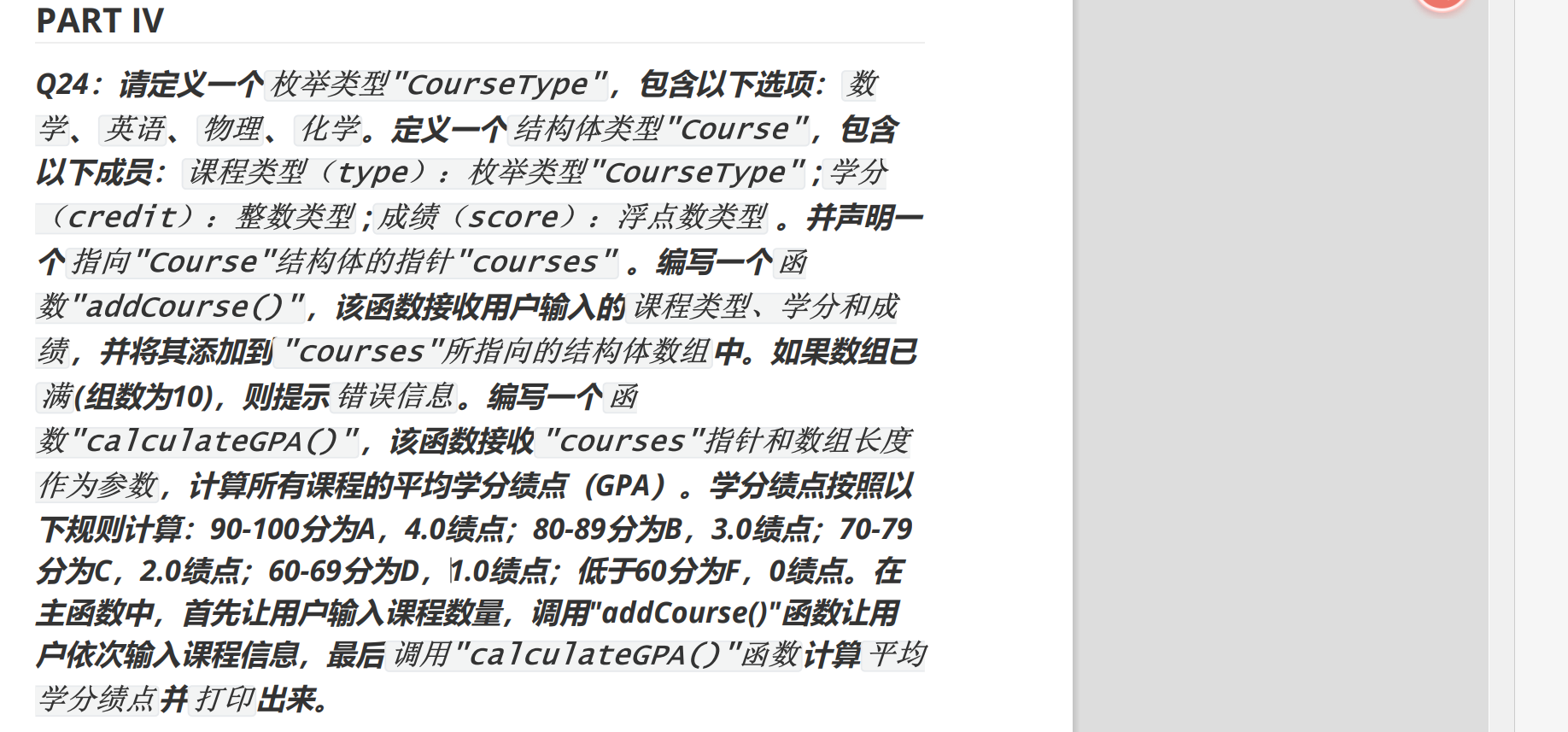
//云台当前状态

gimbalState=GIMBAL\_INIT\_DONE;

printf("云台当前状态：%d",gimbalState);

return 0;

}



24.

#include <stdio.h>

typedef enum

{

Math=0, //数学

English, //英语

Physics, //物理

Chemical //化学

}CourseType;

typedef struct

{

CourseType type; //课程

int credit; //学分

float score; //成绩

}Course;

Course arr[12];

Course \*courses=NULL;

//输入函数

void addCourse(int count)

{

//超过10提示数组已满

if(count>=10)

{

printf("Error:数组已满\n");

return ;

}

//记录科目类型

int type;

printf("请输入科目(0=Math, 1=English, 2=Physics, 3=Chemistry): \n");

scanf("%d",&type);

courses[count].type=(CourseType)type;

printf("请输入学分：\n");

scanf("%d",&courses[count].credit);

printf("请输入成绩：\n");

scanf("%f",&courses[count].score);

}

//计算绩点函数

float calculateGPA(int CourseNumber)

{

//超过10强制等于10

if(CourseNumber>10)

{

CourseNumber=10;

}

float Signal\_GPA=0;

float Sum\_GPA=0;

float Eve\_GPA=0;

int i=0;

//遍历

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

{

if(courses[i].score>=90 && courses[i].score<=90)

Signal\_GPA=4.0;

else if(courses[i].score>=80 && courses[i].score<=89)

Signal\_GPA=3.0;

else if(courses[i].score>=70 && courses[i].score<=79)

Signal\_GPA=2.0;

else if(courses[i].score>=60 && courses[i].score<=69)

Signal\_GPA=1.0;

else if(courses[i].score<60)

Signal\_GPA=0;

Sum\_GPA+=Signal\_GPA;

}

//计算平均数

Eve\_GPA=Sum\_GPA/CourseNumber;

return Eve\_GPA;

}

int main()

{

//令指针指向数组

courses=arr;

//平均绩点

float Eve\_GPA=0;

int i;

//CourseNumber用来计数有多少课程

int CourseNumber=0;

printf("请输入课程数量：");

scanf("%d",&CourseNumber);

printf("请输入课程信息：\n");

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

{

addCourse(i);

}

Eve\_GPA=calculateGPA(CourseNumber);

printf("绩点：%.2f\n",Eve\_GPA);

return 0;

}