***24RoboMaster醒狮电控组梯队考核***

***C语言篇***

# PART I

Q1.以下哪种不是C语言的基本数据类型？ D

A. char B. int C.short D.num

Q2.以下哪个是双精度浮点型？ B

A. int B.double C.num

D.float

Q3.下列哪个是正确的赋值方式？ C

A.boss==1; B.boy>>1; C.girl=1;

D.you&&1;

Q4.下列哪个是错误的运算符？ C

A.&& B.|| C.?? D.!=

Q5. 请问以下关于printf的表达式正确的是 D

1. printf(“this is C program.”\n);
2. printf(‘%c’a);
3. printf(‘%5.2f’,3.1415926);
4. printf(“\n%12.3e”,12346.6788);

Q6.判断:在复合逻辑运算符中,x+=y+1相当于y=x+y+1? A.正确 B

B.错误

Q7.在 int a=5,y=20; a+=(y%(5+a)\*a)/y; 中，若没有逻辑错误，a的值为？ 5

Q8.C语言while和do-while循环的主要区别是？ A.while循环的控制条件比do-while循环控制条件严格 B.do-while的循环体至少无条件执行一次 B

C.do-while允许从外部转到循环体内 D.do-while的循环体不能是复合语句

Q9.下列有关for循环的正确描述是? D

A.for循环只能用于循环次数已经确定的情况 B.for循环是先执行循环体语句后判定表达式 C.在for循环中，不能用break语句跳出循环体

D.for循环体语句中可以包含多条语句，但要用花括号括起来

Q10.break语句和continue语句的区别是什么？

1、break用于跳出整个循环体。

2、continue用于跳出本次循环，开始下一次循环。

**PART II**

***Q11:举例出C语言中的数据类型（越多越好）***

eg: int ; float enum ;

char,doublt,void,unsigned int,long long, short, unsigned long, unsigned char, unsigned short

***Q12:使用*** while ***自定义一个死循环函数***

Alpha

do{}while(1);

***Q13：解释终值***

VAL\_LIMIT

## ，并计算

***、***Beta ***、***Gamma ***的最***

#define MINVALUE -19

#define MAXVALUE 43

#define VAL\_LIMIT(val, min, max) \ do {\

if((val) <= (min))\

{\

(val) = (min);\

}\

else if((val) >= (max))\

{\

(val) = (max);\

}\

} while(0)\

float Alpha = 1.0; float Beta = 76.4; float Gamma = -57.3;

VAL\_LIMIT(Alpha,MINVALUE,MAXVALUE); VAL\_LIMIT(Beta,MINVALUE,MAXVALUE); VAL\_LIMIT(Gamma,MINVALUE,MAXVALUE);

宏定义 VAL\_LIMIT 可以理解为一个限制函数，它的效果是 小过最小取最小，大过最大取最大。由于do-while的特殊性，不管条件成不成立都回执行一次循环体，所以上面都会运行一次循环体，最后结果如下

Alpha = 1.0

Beta = 43.0

Gamma = -19.0

***Q14:于 中表示*** now1 ***与*** now2 ***的***变化曲线

笛卡尔坐标系

float now1 = 50.5 ; float now2 = -30.3 ;

float ref1 = 100.1; float ref2 = -70.7;

float high = 1.0 ; float low = 1.0 ;

void buffer(float \*a, float b, float high\_parameter, float low\_parameter)

{

if (((\*a - b) <= high\_parameter) && ((\*a - b) >= -low\_parameter))

{

\*a = b;

}

else

{

if (\*a < b)

\*a += high\_parameter; if (\*a > b)

\*a -= low\_parameter;

}

}

buffer(&now1,ref1,high,low); buffer(&now2,ref2,high,low);

#include <stdio.h>

// 函数声明

void buffer(float \*a, float b, float high\_parameter, float low\_parameter);

int main()

{

int i;

float value1[100], value2[100];

float now1 = 50.5;

float now2 = -30.3;

float ref1 = 100.1;

float ref2 = -70.7;

float high = 1.0;

float low = 1.0;

printf("Updated now1:\n");

for (i = 0; i < 50; i++)

{

buffer(&now1, ref1, high, low);

value1[i] = now1;

printf("%.1f\t", now1);

}

putchar('\n');

printf("Updated now2:\n");

for (i = 0; i < 50; i++)

{

buffer(&now2, ref2, high, low);

value2[i] = now2;

printf("%.1f\t", now2);

}

return 0;

}

***Q15:请***自定义变量***，并巧妙利用*** ramp\_t ***,*** ramp\_init ***,***

***实现***自定义变量 ***的*** 斜坡式下降***与***斜坡式增长

ramp\_calc

typedef struct

{

uint32\_t get\_count; uint32\_t set\_count; float out;

}ramp\_t;

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

{

ramp->get\_count = 0;

ramp->set\_count = target\_count; ramp->out = 0;

}

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 ramp\_up, ramp\_down;

uint32\_t target\_count\_up = 20;

uint32\_t target\_count\_down = 30;

ramp\_init(&ramp\_up, target\_count\_up);

ramp\_init(&ramp\_down, target\_count\_down);

for (int i = 0; i < ((target\_count\_up >= target\_count\_down) ? target\_count\_up : target\_count\_down); ++i)

{

float result\_up = ramp\_calc(&ramp\_up);

float result\_down = 1 - ramp\_calc(&ramp\_down);

printf("Up: %.2f\tDown: %.2f\n", result\_up, result\_down);

}

return 0;

}

***Q16:现有名为 的时间测量函数,其单位为***

HAL\_GetTick

ms,请利用

***HAL\_GetTick*** 函数返回值的特殊性质,实现程序***1ms*** 定时打

印***"Hello,world !"` ( tip:① 定义变量与函数配合获得∆t值；② if语句)***

//返回值 uwTick 为当前系统运行时间

float HAL\_GetTick(void)

{

return uwTick;

}

#include <stdio.h>

float HAL\_GetTick(void);

void timed\_print\_hello\_world()

{

static float last\_tick = 0.0f;

float current\_tick = HAL\_GetTick();

float delta\_t = current\_tick - last\_tick;

if (delta\_t >= 1.0f)

{

printf("Hello, world !\n");

last\_tick = current\_tick;

}

}

int main()

{

while (1)

{

timed\_print\_hello\_world();

}

return 0;

}

***Q17:使用名为值***

callback

## 的 初始化

alpha ***与 的***

函数指针

beta

#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()

{

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

callback(&alpha, &beta);

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

printf("beta = %f\n", beta);

return 0;

}

***Q18:请你初始化*** alpha ***的值为***gama ***的***低八位***、***beta ***的值为***

gama

***的***高八位

注：uint8\_t 为无符号8位整型

uint16\_t 为无符号32位整型

uint8\_t alpha = 0; uint8\_t beta = 0;

uint16\_t gama = 20001;

uint8\_t alpha = (uint8\_t)(gama & 0xFF);

uint8\_t beta = (uint8\_t)((gama >> 8) & 0xFF);

***∆Q19:下图代码为PID控制算法公式 ,请 ,并利用***

自定义结构体

pid\_t ***、***PID\_Struct\_Init ***、***pid\_calc ***分别初始化*** kp ***、*** ki ***、***kd ***、***maxout ***、***integral\_limit ***的值 , 并得到经*** pid\_calc ***计算后的***最终值***(tip: 可令 kp = 1 ,ki = 0.01,kd = 20***

***,maxout = 500,integral\_limit = 100 并将其传入 PID\_Struct\_Init 进行初始化,最后调用 pid\_calc 得到最终值)`***

enum

{

};

NOW\_ERR = 0, LAST\_ERR, LLAST\_ERR,

typedef struct pid

{

float set; float get; float error[3];

float kp; float ki; float kd;

float pout; float iout; float dout; float out;

int32\_t maxout; int32\_t integral\_limit;

void (\*f\_pid\_init)(struct pid \*pid\_t,

float p, float i, float d,

int32\_t max\_out, int32\_t integral\_limit);

void (\*f\_pid\_reset)(struct pid \*pid\_t,

float p, float i, float d);

}pid\_t;

static void abs\_limit(float \*x,int32\_t limit)

{

if(\*x > limit)

\*x = limit; if(\*x < -limit)

\*x = -limit;

}

static void pid\_init(pid\_t \*pid,float p,float i,float d,int32\_t max\_out,int32\_t integral\_limit)

{

pid->kp = p; pid->ki = i; pid->kd = d;

pid->maxout = max\_out;

pid->integral\_limit = integral\_limit;

}

static void pid\_reset(pid\_t \*pid,float p,float i,float d)

{

pid->kp = p; pid->ki = i; pid->kd = d;

pid->pout = 0;

pid->iout = 0;

pid->dout = 0;

pid->out = 0;

}

void PID\_Struct\_Init(pid\_t \*pid,float p,float i,float d,int32\_t max\_out,int32\_t integral\_limit)

{

pid->f\_pid\_init = pid\_init; pid->f\_pid\_reset = pid\_reset;

pid->f\_pid\_init(pid,p,i,d,max\_out,integral\_limit); pid->f\_pid\_reset(pid,p,i,d);

}

float pid\_calc(pid\_t \*pid,float get,float set)

{

pid->get = get; pid->set = set;

pid->error[NOW\_ERR] = set - get;

pid->pout = pid->kp \* pid->error[NOW\_ERR]; pid->iout += pid->ki \* pid->error[NOW\_ERR];

pid->dout = pid->kd \* (pid->error[NOW\_ERR] - pid->error[LAST\_ERR]);

//积分限幅

abs\_limit(&(pid->iout),pid->integral\_limit); pid->out = pid->pout + pid->iout + pid->dout; abs\_limit(&(pid->out),pid->maxout);

//最终值限幅

pid->out += pid->pout + pid->iout + pid->dout; abs\_limit(&(pid->out), pid->maxout);

pid->error[LLAST\_ERR] = pid->error[LAST\_ERR]; pid->error[LAST\_ERR] = pid->error[NOW\_ERR];

//最终值输出

return pid->out;

}

int main()

{

pid\_t my\_pid;

float kp = 1.0;

float ki = 0.01;

float kd = 20.0;

int32\_t maxout = 500;

int32\_t integral\_limit = 100;

PID\_Struct\_Init(&my\_pid, kp, ki, kd, maxout, integral\_limit);

float setpoint = 100.0;

float current\_value = 80.0;

float result = pid\_calc(&my\_pid, current\_value, setpoint);

printf("PID Result: %.2f\n", result);

return 0;

}

# PART III

***Q20：题目：请自定义一个结构体类型*** "Student" ***，包含以下成员变量***(特殊要求：用结构体与指针相关内容解决)

姓名（字符串类型）学号（整型）

成绩（单精度浮点型）

请编写一个函数，接收一个指向 "Student" 结构体数组的指针，将每个学生的姓名、学号和成绩输出到屏幕上。

#include <stdio.h>

struct Student {

char name[50];

int student\_number;

float score;

};

int main() {

struct Student students[] = {

{"Alice", 1001, 89.5},

{"Bob", 1002, 78.2},

{"Charlie", 1003, 92.0},

};

int numStudents = sizeof(students) / sizeof(students[0]);//算学生人数

printf("学生信息如下:\n");

for (int i = 0; i < numStudents; ++i) {

printf("姓名: %s\n", students[i].name);

printf("学号: %d\n", students[i].student\_number);

printf("成绩: %.2f\n", students[i].score);

printf("\n");

}

return 0;

}

## Q21：请编写一个函数，接收一个整型数组和数组长度作为参数，找出数组中的最大值，并通过指针将最大值返回给调用函数。(特殊要

求:利用结构体与指针相关内容解决)

#include <stdio.h>

int findMax(int \*arr, int length);

int main() {

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

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

//在此填写函数 ↓

int max = findMax(arr, length);

//在此填写函数 ↑

printf("最大值：%d\n", max); return 0;

int findMax(int \*arr, int length)

{

int max = arr[0];

for (int i = 1; i < length; ++i)

{

if (arr[i] > max)

max = arr[i];

}

return max;

}

>注意：已给代码不可更改!!!

***Q22：请编写一个void函数，使其限制电机转速在-4000~4000范围内***

(特殊要求:在不用全局变量的情况下完成 )

#include <stdio.h>

* 注意：不准在这给变量赋值，只可给函数声明！

int main() {

int nowspeed=5000,maxspeed=4000;

//请插入一个void函数 ↓

//请在此插入一个函数 ↑ printf("nowspeed的值",nowspeed)

return 0;

}

//请在此补充函数 void ...

//请在此补充函数↑

* 注意:①不可加入全局变量!

#include <stdio.h>

void limitSpeed(int \*speed);

int main() {

int nowspeed = 5000;

int maxspeed = 4000;

limitSpeed(&nowspeed);

printf("nowspeed的值：%d\n", nowspeed);

return 0;

}

void limitSpeed(int \*speed) {

if (\*speed > 4000) {

\*speed = 4000;

} else if (\*speed < -4000) {

\*speed = -4000;

}

}

②已给代码不可更改!

***Q23：已知 有四种状态，分别为***GIMBAL\_INIT\_NEVER ***，***

云台

GIMBAL\_INIT\_DONE，NO\_ACTION ***，*** IS\_ACTION ***四种，请你定***

***义一个***枚举结构体gimbal\_state\_t ***放入这四种状态；假设***现在***云台状态为***GIMBAL\_INIT\_DONE ***，请你写出程序,使其输出***当前状态***的枚举值***

(特殊要求:利用枚举)

#include <stdio.h>

typedef enum {

GIMBAL\_INIT\_NEVER,

GIMBAL\_INIT\_DONE,

NO\_ACTION,

IS\_ACTION

} gimbal\_state\_t;

int main()

{

gimbal\_state\_t current\_state = GIMBAL\_INIT\_DONE;

switch (current\_state) {

case GIMBAL\_INIT\_NEVER:

printf("当前状态：GIMBAL\_INIT\_NEVER\n");break;

case GIMBAL\_INIT\_DONE:

printf("当前状态：GIMBAL\_INIT\_DONE\n");break;

case NO\_ACTION:

printf("当前状态：NO\_ACTION\n");break;

case IS\_ACTION:

printf("当前状态：IS\_ACTION\n");break;

default:

printf("未知状态\n");

}

return 0;

}

# PART IV

***Q24：请定义一个***枚举类型"CourseType" ***，包含以下选项：*** 数学***、***英语***、***物理***、***化学***。定义一个***结构体类型"Course" ***，包含以下成员：*** 课程类型（type）：枚举类型"CourseType" ***;*** 学分

（credit）：整数类型 ***;*** 成绩（score）：浮点数类型 ***。并声明一个***指向"Course"结构体的指针"courses" ***。编写一个***函

数"addCourse()" ***，该函数接收用户输入的***课程类型、学分和成绩***，并将其添加到***"courses"所指向的结构体数组***中。如果数组已***满***(组数为10)，则提示***错误信息***。编写一个***函

数"calculateGPA()" ***，该函数接收***"courses"指针和数组长度 作为参数***，计算所有课程的平均学分绩点（GPA）。学分绩点按照以下规则计算：90-100分为A，4.0绩点；80-89分为B，3.0绩点；70-79分为C，2.0绩点；60-69分为D，1.0绩点；低于60分为F，0绩点。在主函数中，首先让用户输入课程数量，调用"addCourse()"函数让用户依次输入课程信息，最后***调用"calculateGPA()"函数***计算***平均学分绩点***并***打印***出来。***

//

#include <iostream>

using namespace std;

// 枚举类型定义

enum CourseType {

数学,

英语,

物理,

化学

};

// 结构体类型定义

struct Course {

CourseType type; // 课程类型

int credit; // 学分

float score; // 成绩

};

// 函数声明

void addCourse(Course\* courses, int& count);

float calculateGPA(Course\* courses, int count);

int main() {

const int MAX\_COURSES = 10; // 最大课程数量

Course courses[MAX\_COURSES]; // 结构体数组

int courseCount; // 用户输入的课程数量

// 获取课程数量

cout << "请输入课程数量（最多10门课程）：";

cin >> courseCount;

// 检查输入是否有效

if (courseCount < 0 || courseCount > MAX\_COURSES) {

cout << "无效的课程数量！请重新运行程序。\n";

return 1;

}

// 添加课程信息

for (int i = 0; i < courseCount; ++i) {

cout << "请输入第 " << i + 1 << " 门课程信息：" << endl;

addCourse(courses, i);

}

// 计算并输出平均学分绩点

float gpa = calculateGPA(courses, courseCount);

cout << "平均学分绩点（GPA）为：" << gpa << endl;

return 0;

}

// 添加课程函数

void addCourse(Course\* courses, int& count) {

int typeIndex;

cout << "课程类型（0: 数学, 1: 英语, 2: 物理, 3: 化学）：";

cin >> typeIndex;

// 输入验证

if (typeIndex < 0 || typeIndex > 3) {

cout << "无效的课程类型！请重新输入。\n";

addCourse(courses, count); // 递归调用重新输入

return;

}

courses[count].type = static\_cast<CourseType>(typeIndex);

cout << "学分：";

cin >> courses[count].credit;

cout << "成绩：";

cin >> courses[count].score;

count++; // 数组下标移动到下一个位置

}

// 计算平均学分绩点函数

float calculateGPA(Course\* courses, int count) {

float totalCredits = 0.0;

float totalGradePoints = 0.0;

for (int i = 0; i < count; ++i) {

// 计算绩点

float gradePoint;

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

gradePoint = 4.0;

} else if (courses[i].score >= 80) {

gradePoint = 3.0;

} else if (courses[i].score >= 70) {

gradePoint = 2.0;

} else if (courses[i].score >= 60) {

gradePoint = 1.0;

} else {

gradePoint = 0.0;

}

// 计算加权绩点

totalCredits += courses[i].credit;

totalGradePoints += gradePoint \* courses[i].credit;

}

// 计算平均绩点

if (totalCredits > 0) {

return totalGradePoints / totalCredits;

} else {

return 0.0;

}

}

***@醒狮机器人实验室电控组***