Matthew Hulme 103112063

1

a

Int a=1, b=2, \*ptr;

ptr=&b;

|  |  |  |  |
| --- | --- | --- | --- |
| **Statement** | **a(1000)** | **b(2000)** | **ptr** |
| Int a=1, b=2, \*ptr; | 1 | 2 | ? |
| ptr=&b; | 1 | 2 | 2000 |

b

int a=1, b=2, \*ptr=&b;

a=\*ptr;

|  |  |  |  |
| --- | --- | --- | --- |
| **Statement** | **a(1000)** | **b(2000)** | **ptr** |
| int a=1, b=2, \*ptr=&b; | 1 | 2 | 2000 |
| a=\*ptr; | 2 | 2 | 2000 |

c

int a=1, b=2, c=5, \*ptr=&c;

b=\*ptr;

\*ptr=a;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Statement** | **a(1000)** | **b(2000)** | **c(3000)** | **ptr** |
| int a=1, b=2, c=5, \*ptr=&c; | 1 | 2 | 5 | 3000 |
| b=\*ptr; | 1 | 5 | 5 | 3000 |
| \*ptr=a; | 1 | 5 | 1 | 3000 |

d

int a=1, b=2, c=5, \*ptr;

ptr=&c;

c=b;

a=\*ptr;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Statement** | **a(1000)** | **b(2000)** | **c(3000)** | **ptr** |
| int a=1, b=2, c=5, \*ptr; | 1 | 2 | 5 | ? |
| ptr=&c; | 1 | 2 | 5 | 3000 |
| c=b; | 1 | 2 | 2 | 3000 |
| a=\*ptr; | 2 | 2 | 2 | 3000 |

e

double x=15.6, y=10.2, \*ptr\_1=&y, \*ptr\_2=&x;

\*ptr\_1 = \*ptr\_2 + x;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Statement** | **x(1000)** | **y(2000)** | **ptr\_1** | **ptr\_2** |
| double x=15.6, y=10.2, \*ptr\_1=&y, \*ptr\_2=&x; | 15.6 | 10.2 | 2000 | 1000 |
| \*ptr\_1 = \*ptr\_2 + x; | 15.6 | 31.2 | 2000 | 1000 |

f

int w=10, x=2, \*ptr\_2=&x;

\*ptr\_2 -= w;

|  |  |  |  |
| --- | --- | --- | --- |
| **Statement** | **w(1000)** | **x(2000)** | **ptr\_2** |
| int w=10, x=2, \*ptr\_2=&x; | 10 | 2 | 2000 |
| \*ptr\_2 -= w; | 10 | -8 | 2000 |

2

int g[]={2,4,5,8,10,32,78};

int \*ptr1=&g[0], \*ptr2=&g[3];

|  |  |  |  |
| --- | --- | --- | --- |
| **Statement** | **g(1000)** | **ptr1** | **ptr2** |
| int g[]={2,4,5,8,10,32,78}; | 2,4,5,8,10,32,78 | ? | ? |
| int \*ptr1=&g[0], \*ptr2=&g[3]; | 2,4,5,8,10,32,78 | 1000 | 1000 + sizeof(int) \* 3 |

|  |  |
| --- | --- |
| **Identifier** | **Value** |
| \*g | 2 |
| \*(g+1) | 4 |
| \*g+1 | 3 |
| \*(g+5) | 32 |
| \*ptr1 | 2 |
| \*ptr2 | 8 |
| \*(ptr1+1) | 4 |
| \*(ptr2+2) | 32 |

3

int x[2][4]={{1,8,7,6},{2,4,-1,0}}, \*xptr=&x[0][0];

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 8 | 7 | 6 |
| 2 | 4 | -1 | 0 |

1. \*xptr == 1
2. \*(xptr+2) == 7
3. \*xptr + 2 == 3
4. \*(xptr+1) + \*(xptr+3) == 14

4

Peter

55

66

44

67

Lilly

100

90

43

89

John

34

56

78

65

Mary

45

56

78

65

Mary

45

56

78

90

Alex

30

45

65

54

struct student

{

char name[10];

int marks[4];

};

Take input from a file, then fill it into the struct.

Calculate the average through a user defined function.

Fail the student if any mark is below 60.

COS10007 Week 1 Question 4

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#include <stdlib.h>

#include <unistd.h>

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

/\* Parse an input file for student marks and fill them into a struct,

\* then fail them if they score below 60 on any subject. \*/

const size\_t ammount\_of\_students = 5;

/\* Struct for storing students. \*/

struct student {

char name[10];

int marks[4];

};

/\* Print all arguments for debug purposes. \*/

printf("Arguments:\n");

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

printf("%d : \"%s\"\n", i + 1, argv[i]);

}

printf("\n");

/\*\*/

/\* Parse file for student names and marks. \*/

void parseStudentDetails( FILE \* file\_pointer,

size\_t stud\_len,

struct student stud[]) {

const size\_t student\_mark\_len = 4;

/\* I'm just gonna be lazy and assume that the formatting is:

\* Name1

\* Mark1

\* Mark2

\* Mark3

\* Mark4

\* Name2...

\*/

char line[10];

for(int s = 0; s < stud\_len; s++) {

fscanf(file\_pointer, "%s", &line);

//printf("%s\n", line);

strcpy(stud[s].name, line);

for(int i = 0; i < student\_mark\_len; i++) {

fscanf(file\_pointer, "%s", &line);

//printf("%s\n", line);

stud[s].marks[i] = atoi(line);

}

}

}

void checkStudentMarks(size\_t stud\_len, struct student stud[]) {

for(int i = 0; i < stud\_len; i++) {

bool pass = true;

printf("Name: %s\n", stud[i].name);

for(int a = 0; a < 4; a++) {

printf("Mark: %d", stud[i].marks[a]);

if(stud[i].marks[a] < 60) {

printf(" (fail)");

pass = false;

}

printf("\n");

}

printf("%s", stud[i].name);

if(pass == true) {

printf(" has passed all subjects.\n");

} else {

printf(" has failed at least one subject.\n");

}

printf("\n");

}

printf("\n");

}

/\* Check there is only argument then open the file. \*/

FILE \* fp = NULL;

if(argc == 2) {

if((fp = fopen(argv[1], "r")) == NULL) {

printf("File failed to open.");

return 1;

} else {

struct student Students[ammount\_of\_students];

parseStudentDetails(fp,

sizeof(Students) / sizeof(\*Students),

Students);

checkStudentMarks( sizeof(Students) / sizeof(\*Students),

Students);

fclose(fp);

}

} else {

printf("One argument accepted, no more, no less.\n");

return 1;

}

return 0;

}

