

Sprint 08

Marathon C

April 15, 2020



ucode

Contents



Engage	2
Investigate	3
Act: Task 00 > Header Intro	5
Act: Task 01 > Structure Intro	6
Act: Task 02 > Decimal to hex	7
Act: Task 03 > Hex to decimal	8
Act: Task 04 > Get address	9
Act: Task 05 > Neo's choice	10
Act: Task 06 > Matrix need a new agent	12
Act: Task 07 > More agents!!!	13
Act: Task 08 > Ex-ter-mi-nate agents	14
Act: Task 09 > Smiths	15
Share	16

Engage



DESCRIPTION

Hello!

We hope that yesterday's **Sprint** helped you understand how memory works. And now this information has a place in your mind palace.

As you have already noticed, the more complex the task, the more knowledge it requires. The same applies to data manipulation. Standard data types are not enough to solve the problem effectively. The number of files, functions, and variables is becoming greater and greater. The number of different elements increases, and it becomes more difficult to manage them.

Today you must deal with this issue.

This **Sprint** will help you find solutions to many issues you've already faced before. You will learn what are **headers** and **structures**.

BIG IDEA

Problem-solving using effective tools.

ESSENTIAL QUESTION

How to properly manage data in a program?

CHALLENGE

Learn headers and structures.

Investigate



GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What data types exist in C?
 - How to create user defined data types in C?
 - What are the primary data types?
 - What are the derived data types?
 - What are arrays?
- What is a structure?
 - How to declare structure variables?
 - How to initialize structure members?
 - How to access structure elements?
 - What is designated initialization?
- What are headers?
 - How to include headers in a file?
 - How to use the created header file?
 - How to compile a program using a header file?
 - What are once-only headers? How to use `#pragma once`?
- What is a positional number system?
 - What types exist?
 - What is the base of a numeral system?
 - What is the `0x` prefix for hexadecimal numbers?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Repeat the basics from yesterday. Repeat everything you know and do not know about pointers. Today they are also needed.
- Find information about the terms: `positional number system`, `header file`, `typedef`, `struct` in C.
- Read the tasks below.
- Clone your git repository that is issued on the challenge page in the LMS.
- Open the **Auditor** and find examples of correct use of header files and structures.
- Arrange to brainstorm tasks with other students.
- Try to implement your thoughts in code.



ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
- Analyze all information you have collected during the preparation stages.
- Perform only those tasks that are given in this document.
- Submit your files using the layout described in the story. Only useful files allowed, garbage shall not pass!
- Compile C-files with clang compiler and use these flags:
`clang -std=c11 -Wall -Wextra -Werror -Wpedantic .`
- Your program must manage memory allocations correctly. A memory that is no longer needed must be freed, otherwise, the task is considered incomplete.
- Pay attention to what is allowed in a certain task. Use of forbidden stuff is considered a cheat and your tasks will be failed.
- Complete tasks according to the rules specified in the `Auditor`.
- The solution will be checked and graded by students like you. `Peer-to-Peer learning`.
- Also, the challenge will pass automatic evaluation which is called `Oracle`.
- If you have any questions or don't understand something, ask other students or just Google it.
- Use your brain and follow the white rabbit to prove that you are the Chosen one!

Act: Task 00



NAME

Header intro

DIRECTORY

t00/

SUBMIT

header.h

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a header that contains prototypes of your library functions. Prototypes of the next functions are mandatory:

- `mx_printchar`
- `mx_printint`
- `mx_printstr`
- `mx_strcpy`
- `mx_strlen`
- `mx_strcmp`
- `mx_isdigit`
- `mx_isspace`
- `mx_atoi`

Oracle will test these functions by using your header.

The headers must also comply with the Auditor rules. Starting from this challenge:

- typedef/struct/union/enum must only be declared in header files unless specified otherwise
- function prototypes and macro definitions must be only in header files

SEE ALSO

Once-Only Headers
C - Header Files

NAME

DIRECTORY

t01/

SUBMIT

duplicate.h, mx_del_dup_sarr.c, mx_copy_int_arr.c

ALLOWED FUNCTIONS

malloc, free

DESCRIPTION

mx_del_dup_arr

s_intarr

RETURN

NULL

SYNOPSIS

```
typedef struct s_intarr
{
    int *arr;
    int size;
} t_intarr;
```

```
t_intarr *mx_del_dup_sarr(t_intarr *src);
```

SEE ALSO

C struct
C programming structure

```
                                //returns "34"  
mx_nbr_to_hex(1000); //returns "3e8"
```


NAME

DIRECTORY

```
t03/
```

SUBMIT

```
hex_to_nbr.h, mx_hex_to_nbr.c, mx_isdigit.c, mx_isalpha.c, mx_islower.c, mx_isupper.c
```

ALLOWED FUNCTIONS

None

DESCRIPTION

```
unsigned long
```

RETURN

```
unsigned long
```

SYNOPSIS

```
unsigned long mx_hex_to_nbr(const char *hex);
```

EXAMPLE

```
mx_hex_to_nbr("C4"); //returns 196
mx_hex_to_nbr("FADE"); //returns 64222
mx_hex_to_nbr("ffffffffffff"); //returns 281474976710655
```

Act: Task 04



NAME

Get address

DIRECTORY

```
t04/
```

SUBMIT

```
get_address.h, mx_get_address.c, mx_nbr_to_hex.c, mx_strcpy.c, mx_strlen.c, mx_strnew.c
```

ALLOWED FUNCTIONS

```
malloc, free
```

DESCRIPTION

Create a function that takes a pointer and returns its address in memory in a hexadecimal format with the prefix `"0x"`.

RETURN

Returns the address of the pointer as a string.

SYNOPSIS

```
char *mx_get_address(void *p);
```

Act: Task 05

NAME

Neo's choice

DIRECTORY

t05/

SUBMIT

choice.h

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a header file `choice.h` with which the program compiles and works.
If Neo chooses:

- the red pill, the program prints `"Follow me!"`
- the blue pill, the program prints `"Perhaps I was wrong about you, Neo."`
- something else, the program prints `"Are you sure about that?"`

SYNOPSIS

```
#include "choice.h"

t_phrase *choice(int pill) {
    char *res;

    if (pill == MX_RED_PILL) {
        res = mx_strdup(MX_SUCCESS_PHRASE);
    }
    else if (pill == MX_BLUE_PILL) {
        res = mx_strdup(MX_FAIL_PHRASE);
    }
    else {
        res = mx_strdup(MX_UNDEFINED_PHRASE);
    }
    return res;
}

int main(void) {
    t_phrase *phrase1 = choice(MX_RED_PILL);
    t_phrase *phrase2 = choice(MX_BLUE_PILL);
    t_phrase *phrase3 = choice((MX_RED_PILL + MX_BLUE_PILL) * 2);

    printf("%s\n", phrase1);
    printf("%s\n", phrase2);
```



```
printf("%s\n", phrase3);  
return 0;  
}
```

Act: Task 06

NAME

Matrix need a new agent

DIRECTORY

t06/

SUBMIT

create_agent.h, mx_create_agent.c, mx_strdup.c, mx_strnew.c, mx_strlen.c, mx_strcpy.c

ALLOWED FUNCTIONS

malloc

DESCRIPTION

The Matrix chose you. In this task you must:

- develop a new agent creator
- include `agent.h` in your header file to be able to use the structure `s_agent`. The Matrix will use your function with its own header `agent.h`
- create a function that allocates new (duplicate) memory for the `name` parameter

RETURN

- returns the `pointer` to the allocated memory of the new structure
- returns `NULL` if the `name` is `NULL` or agent creation fails

SYNOPSIS

```
typedef struct s_agent
{
    char *name;
    int power;
    int strength;
} t_agent;
```

```
t_agent *mx_create_agent(char *name, int power, int strength);
```

EXAMPLE

```
agent = mx_create_agent("Smith", 150, 66);
//agent->name is "Smith"
//agent->power is 150
//agent->strength is 66
```

Act: Task 07

NAME

More agents!!!

DIRECTORY

t07/

SUBMIT

create_new_agents.h, mx_create_new_agents.c, mx_create_agent.c, mx_strdup.c, mx_strnew.c, mx_strlen.c, mx_strcpy.c

ALLOWED FUNCTIONS

malloc, free

DESCRIPTION

In this task, you must follow the next items.

- Create a function that makes a `NULL-terminated` array of pointers to agents
- Data for each agent are stored in 3 different arrays: `name`, `power` and `strength`
- Each agent characteristic is placed with the same index in the respective array
- You must use the structure from `agent.h`

RETURN

- returns the `NULL`-terminated array of agents
- returns `NULL` if one of the parameters of the function is `NULL` or agent creation fails

SYNOPSIS

```
t_agent **mx_create_new_agents(char **name, int *power, int *strength, int count);
```

EXAMPLE

```
names = {"Thompson", "Smith", "Colson"};
powers = {33, 66, 99};
strengths = {133, 166, 196};
mx_create_new_agents(names, powers, strengths, 3); //returns 't_agent' type array
```

Act: Task 08



NAME

Ex-ter-mi-nate agents

DIRECTORY

t08/

SUBMIT

exterminate_agents.h, mx_exterminate_agents.c

ALLOWED FUNCTIONS

free

DESCRIPTION

Create a function that:

- frees the `NULL-terminated` array of agents
- frees the contents of each agent
- sets a pointer to `NULL`
- uses the structure from `agent.h`

SYNOPSIS

```
void mx_exterminate_agents(t_agent ***agents);
```

Act: Task 09

NAME

Smiths

DIRECTORY

t09/

SUBMIT

only_smiths.h, mx_only_smiths.c, mx_strcmp.c, mx_terminate_agents.c, mx_create_agent.c, mx_strdup.c, mx_strnew.c, mx_strlen.c, mx_strcpy.c

ALLOWED FUNCTIONS

malloc, free

DESCRIPTION

In this task, you must follow the next directions.

- Create a function that creates a new **NULL-terminated** array of pointers to agents
- The new array has only agents with the name **Smith** and a strength lower than the **strength** parameter of the function
- Input **agents** must be exterminated
- You must use the structure from **agent.h**

RETURN

- returns the new filtered array
- returns **NULL** if the original array is **NULL** or the new array creation fails

SYNOPSIS

```
t_agent **mx_only_smiths(t_agent **agents, int strength);
```

EXAMPLE

```
agents[0] = mx_create_agent("Smith", 150, 166);
agents[1] = mx_create_agent("Brown", 147, 57);
agents[2] = mx_create_agent("Smith", 151, 65);
agents[3] = mx_create_agent("Smith", 123, 321);
agents[4] = NULL;
mx_only_smiths(agents, 100); //returns array with 1 element
```