Version: 1.0

Selection

C-Memory-Management

Summary

Implement functions to manage memory dynamically in C, including allocation, string manipulation, and base conversion.

#C #Memory #DynamicAllocation

42

Intellectual Property Disclaimer

All content presented in this training module, including but not limited to texts, images, graphics, and other materials, is protected by intellectual property rights held by Association 42.

Terms of Use:

- **Personal use:** You are permitted to use the contents of this module solely for personal purpose. Any commercial use, reproduction, distribution, modification, or public display is strictly prohibited without prior written permission from Association 42.
- **Respect for Integrity:** You must not alter, transform, or adapt the content in any way that could harm its integrity.

Protection of Rights:

Any violation of these terms constitutes an infringement of intellectual property rights and may result in legal action. We reserve the right to take all necessary measures to protect our rights, including but not limited to claims for damages.

For any questions regarding the use of the content or to obtain authorization, please contact: `legal@42.fr`

Contents

1 Instructions 1

2 Foreword 5

3 Exercise 0: ft_strdup 7

4 Exercise 1: ft_range 8

5 Exercise 2: ft_ultimate_range 9

6 Exercice 3: ft_strjoin 10

7 Exercise 4: ft_convert_base 11

8 Exercise 5: ft_split 13

9 Submission and peer-evaluation 14

Chapter 1

Instructions

- Only this page will serve as a reference: do not trust rumors.
- Watch out! This document could potentially change up until submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- Additionally, your exercises will be checked and graded by a program called Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated, and there is no way to negotiate with it. So, to avoid bad surprises, be as thorough as possible.
- Moulinette is not very open-minded. It won't try to understand your code if it doesn't adhere to the Norm. Moulinette relies on a program called `norminette` to check if your files respect the norm. TL;DR: it would be foolish to submit work that doesn't pass `norminette`'s check.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` consider a successfully completed harder exercise if an easier one is not perfectly functional.
- Using a forbidden function is considered cheating. Cheaters get -42, and this grade is non-negotiable.
- You'll only have to submit a main() function if we ask for a program.
- Moulinette compiles with these flags: -Wall -Wextra -Werror, and uses `cc`.
- If your program doesn't compile, you'll get `0`.
- You cannot leave any additional files in your directory other than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / man / the Internet / ...`.
- Check out the Slack Piscine.

- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- By Odin, by Thor! Use your brain!!!

Do not forget to add the *standard 42 header* in each of your .c/.h files. Norminette checks its existence anyway!

Norminette must be launched with the *-R CheckForbiddenSourceHeader* flag. Moulinette will use it too.

Context

The C Piscine is intense. It's your first big challenge at 42 — a deep dive into problem-solving, autonomy, and community.

During this phase, your main objective is to build your foundation — through struggle, repetition, and especially **peer-learning** exchange.

In the AI era, shortcuts are easy to find. However, it's important to consider whether your AI usage is truly helping you grow — or simply getting in the way of developing real skills.

The Piscine is also a human experience — and for now, nothing can replace that. Not even AI.

For a more complete overview of our stance on AI — as a learning tool, as part of the ICT curriculum, and as a growing expectation in the job market — please refer to the dedicated FAQ available on the intranet.

Main message

- 👉 Build strong foundations without shortcuts.
- 👉 Really develop tech & power skills.
- 👉 Experience real peer-learning, start learning how to learn and solve new problems.
- 👉 The learning journey is more important than the result.
- 👉 Learn about the risks associated with AI, and develop effective control practices and countermeasures to avoid common pitfalls.

Learner rules:

- You should apply reasoning to your assigned tasks, especially before turning to AI.
- You should not ask for direct answers to the AI.
- You should learn about 42 global approach on AI.

Phase outcomes:

Within this foundational phase, you will get the following outcomes:

- Get proper tech and coding foundations.
- Know why and how AI can be dangerous during this phase.

Comments and example:

- Yes, we know AI exists — and yes, it can solve your activities. But you're here to learn, not to prove that AI has learned. Don't waste your time (or ours) just to demonstrate that AI can solve the given problem.
- Learning at 42 isn't about knowing the answer — it's about developing the ability to find one. AI gives you the answer directly, but that prevents you from building your own reasoning. And reasoning takes time, effort, and involves failure. The path to success is not supposed to be easy.
- Keep in mind that during exams, AI is not available — no internet, no smartphones, etc. You'll quickly realise if you've relied too heavily on AI in your learning process.
- Peer learning exposes you to different ideas and approaches, improving your interpersonal skills and your ability to think divergently. That's far more valuable than just chatting with a bot. So don't be shy — talk, ask questions, and learn together!
- Yes, AI will be part of the curriculum — both as a learning tool and as a topic in itself. You'll even have the chance to build your own AI software. In order to learn more about our crescendo approach you'll go through in the documentation available on the intranet.

✓ Good practice:

> I'm stuck on a new concept. I ask someone nearby how they approached it. We talk for 10 minutes — and suddenly it clicks. I get it.

✗ Bad practice:

> I secretly use AI, copy some code that looks right. During peer evaluation, I can't explain anything. I fail. During the exam — no AI — I'm stuck again. I fail.

Chapter 2

Foreword

Here are some lyrics extract from the `Harry Potter` saga:

```
Oh you may not think me pretty,
But don't judge on what you see,
I'll eat myself if you can find
A smarter hat than me.

You can keep your bowlers black,
Your top hats sleek and tall,
For I'm the Hogwarts Sorting Hat
And I can cap them all.

The Sorting Hat, stored in the Headmaster's Office.
There's nothing hidden in your head
The Sorting Hat can't see,
So try me on and I will tell you
Where you ought to be.

You might belong in Gryffindor,
Where dwell the brave at heart,
Their daring, nerve, and chivalry
Set Gryffindors apart;

You might belong in Hufflepuff,
Where they are just and loyal,
Those patient Hufflepuffs are true
And unafraid of toil;

Or yet in wise old Ravenclaw,
If you've a ready mind,
Where those of wit and learning,
Will always find their kind;

Or perhaps in Slytherin
You'll make your real friends,
Those cunning folks use any means
To achieve their ends.
```

So put me on! Don’t be afraid!
And don’t get in a flap!
You’re in safe hands (though I have none)
For I’m a Thinking Cap!

Unfortunately, this subject’s got nothing to do with the Harry Potter saga, which is too bad, because your exercises won’t be done by magic.

Chapter 3

Exercise 0: ft_strdup

	Exercise: 0	
ft_strdup		
Directory: *ex0/*		
Files to Submit: `ft_strdup.c`		
Authorized: `malloc`		

- Reproduce the behavior of the function `strdup` (man strdup).
- Here's how it should be prototyped :

Prototype:

```
char *ft_strdup(char *src);
```

Chapter 4

Exercise 1: ft_range

	Exercise: 1	
ft_range		
Directory: *ex1/*		
Files to Submit: `ft_range.c`		
Authorized: `malloc`		

- Create a function `ft_range` which returns an array of `ints`. This `int` array should contain all values between `min` and `max`.
- `Min` included - `max` excluded.
- Here's how it should be prototyped :

Prototype:

```
int *ft_range(int min, int max);
```

- If `min`'s value is greater or equal to `max`'s value, a null pointer should be returned.

Chapter 5

Exercise 2: ft_ultimate_range

	Exercise: 2	
ft_ultimate_range		
Directory: *ex2/*		
Files to Submit: `ft_ultimate_range.c`		
Authorized: `malloc`		

- Create a function `ft_ultimate_range` which allocates and assigns an array of `ints`. This `int` array should contain all values between `min` and `max`.
- `Min` included - `max` excluded.
- Here's how it should be prototyped :

Prototype:

```
int ft_ultimate_range(int **range, int min, int max);
```

- The size of `range` should be returned (or -1 on error).
- If the value of `min` is greater or equal to `max`'s value, `range` will point to NULL and it should return 0.

Chapter 6

Exercice 3: ft_strjoin

	Exercise: 3	
ft_strjoin		
Directory: *ex3/*		
Files to Submit: `ft_strjoin.c`		
Authorized: `malloc`		

- Write a function that will concatenate all the strings pointed by `strs` separated by `sep`.
- `size` is the number of strings in `strs`
- if `size` is 0, you must return an empty string that you can free().

Prototype:

```
char *ft_strjoin(int size, char **strs, char *sep);
```

Chapter 7

Exercise 4: ft_convert_base

	Exercise: 4	
ft_convert_base		
Directory: *ex4/*		
Files to Submit: `ft_convert_base.c, ft_convert_base2.c`		
Authorized: `malloc, free`		

- Create a function that returns the result of the conversion of the string `nbr` from a base `base_from` to a base `base_to`.

  

  The base system consists of all the symbols used to represent the number.

  - `0123456789` is the commonly used base system for representing decimal numbers.
  - `01` is a binary base system.
  - `0123456789ABCDEF` is a hexadecimal base system.
  - `poneyvif` is an octal base system.

- `nbr`, `base_from`, `base_to` may be not writable.
- `nbr` is in a specific base, given as a second parameter.
- The number represented by `nbr` must fit inside an `int`.
- If a base is wrong, NULL should be returned. Examples of invalid arguments:
  - The base is empty or has only one character.
  - The base contains duplicate characters.
  - The base contains '+', '-', or whitespace characters.
- The returned number must be prefixed only by a single and unique '-' if necessary, no whitespaces, no '+'.

- Here's how it should be prototyped :

Prototype:

```
char *ft_convert_base(char *nbr, char *base_from, char
*base_to);
```

Chapter 8

Exercise 5: ft_split

	Exercise: 5	
ft_split		
Directory: *ex*5/		
Files to Submit: `ft_split.c`		
Authorized: `malloc`		

- Create a function that splits a string of characters depending on another string of characters.
- You'll have to use each character from the string `charset` as a separator.
- The function returns an array where each element of the array contains the address of a string wrapped between two separators. The last element of that array should equal to 0 to indicate the end of the array.
- There cannot be any empty strings in your array. Get your own conclusions accordingly.
- The string given as an argument won't be modifiable.
- Here's how it should be prototyped :

Prototype:

```
char **ft_split(char *str, char *charset);
```

Chapter 9

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.

You need to return only the files requested by the subject of this project.