0x04 more functions nested loops

README

**C - More functions, more nested loops**

[**0. isupper**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/0-isupper.c)

Write a function that checks for uppercase character.

* Prototype: int isupper(int c);
* Returns 1 if c is uppercase
* Returns 0 otherwise

[**1. isdigit**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/1-isdigit.c)

Write a function that checks for a digit (0 through 9).

* Prototype: int isdigit(int c);
* Returns 1 if c is a digit
* Returns 0 otherwise

[**2. Collaboration is multiplication**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/2-mul.c)

Write a function that multiplies two integers.

* Prototype: int mul(int a, int b);

[**3. The numbers speak for themselves**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/3-print_numbers.c)

Write a function that prints the numbers, from 0 to 9, followed by a new line.

* Prototype: void print\_numbers(void);
* You can only use \_putchar twice in your code

[**4. I believe in numbers and signs**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/4-printmostnumbers.c)

Write a function that prints the numbers, from 0 to 9, followed by a new line.

* Prototype: void print\_most\_numbers(void);
* Do not print 2 and 4
* You can only use \_putchar twice in your code

[**5. Numbers constitute the only universal language**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/5-morenumbers.c)

Write a function that prints 10 times the numbers, from 0 to 14, followed by a new line.

* Prototype: void more\_numbers(void);
* You can only use \_putchar three times in your code

[**6. The shortest distance between two points is a straight line**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/6-printline.c)

Write a function that draws a straight line in the terminal.

* Prototype: void print\_line(int n);
* You can only use \_putchar function to print
* Where n is the number of times the character \_ should be printed
* The line should end with a \n
* If n is 0 or less, the function should only print \n

[**7. I feel like I am diagonally parked in a parallel universe**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/7-printdiagonal.c)

Write a function that draws a diagonal line on the terminal.

* Prototype: void print\_diagonal(int n);
* You can only use \_putchar function to print
* Where n is the number of times the character \ should be printed
* The diagonal should end with a \n
* If n is 0 or less, the function should only print a \n

[**8. You are so much sunshine in every square inch**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/8-printsquare.c)

Write a function that prints a square, followed by a new line.

* Prototype: void print\_square(int size);
* You can only use \_putchar function to print
* Where size is the size of the square
* If size is 0 or less, the function should print only a new line
* Use the character # to print the square

[**9. Fizz-Buzz**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/9-fizzbuzz.c)

The “Fizz-Buzz test” is an interview question designed to help filter out the 99.5% of programming job candidates who can’t seem to program their way out of a wet paper bag.

\*Write a program that prints the numbers from 1 to 100, followed by a new line. But for multiples of three print Fizz instead of the number and for the multiples of five print Buzz. For numbers which are multiples of both three and five print FizzBuzz.

* Each number or word should be separated by a space
* You are allowed to use the standard library

[**10. Triangles**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/10-printtriangle.c)

Write a function that prints a triangle, followed by a new line.

* Prototype: void print\_triangle(int size);
* You can only use \_putchar function to print
* Where size is the size of the triangle
* If size is 0 or less, the function should print only a new line
* Use the character # to print the triangle

[**11. The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/100-primefactor.c)

The prime factors of 1231952 are 2, 2, 2, 2, 37 and 2081.

Write a program that finds and prints the largest prime factor of the number 612852475143, followed by a new line.

* You are allowed to use the standard library
* Your program will be compiled with this command: gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100-prime\_factor.c -o 100-prime\_factor -lm

[**12. Numbers have life; they're not just symbols on paper**](https://github.com/KipronohVincent/alx-low_level_programming/blob/master/0x04-more_functions_nested_loops/101-print_number.c)

Write a function that prints an integer.

* Prototype: void print\_number(int n);
* You can only use \_putchar function to print
* You are not allowed to use long
* You are not allowed to use arrays or pointers
* You are not allowed to hard-code special values

===================================================

MAIN.H CODE

#ifndef MAIN\_H

#define MAIN\_H

int \_putchar(char c);

int \_isupper(int c);

int \_isdigit(int c);

int mul(int a, int b);

void print\_numbers(void);

void print\_most\_numbers(void);

void more\_numbers(void);

void print\_line(int n);

void print\_diagonal(int n);

void print\_square(int size);

void print\_triangle(int size);

void print\_number(int n);

#endif

===================================================

\_PUTCHAR CODE

#include "main.h"

#include <unistd.h>

/\*\*

 \* \_putchar - writes the character c to stdout

 \* @c: The character to print

 \*

 \* Return: On success 1.

 \* On error, -1 is returned, and errno is set appropriately.

 \*/

int \_putchar(char c)

{

 return (write(1, &c, 1));

}

===================================================

0-isupper.c CODE

#include "main.h"

/\*\*

\* \_isupper - checks for uppercase character

\* @c: Variable text

\* Return: Always 0.

\*/

int \_isupper(int c)

{

 if (c >= 'A' && c <= 'Z')

 {

 return (1);

 }

 else

{

 return (0);

}

}

==================================================

1-isdigit.c CODE

#include "main.h"

/\*\*

 \* \_isdigit - checks for a digit (0 through 9)

 \* @c: int to be checked

 \* Return: 1 if c is a digit, 0 otherwise

 \*/

int \_isdigit(int c)

{

 return (c >= '0' && c <= '9');

}

===========================================

2-mul.c CODE

#include "main.h"

/\*\*

 \* mul - checks for checks for a digit (0 through 9).

 \* @a: a - Variable

 \* @b: b - variable

 \* Return: Always 0.

 \*/

int mul(int a, int b)

{

 int mul;

 mul = a \* b;

 return (mul);

}

================================================

3-print\_numbers.c CODE

#include "main.h"

/\*\*

 \* print\_numbers - checks for checks for a digit (0 through 9).

 \*

 \* Return: Always 0.

 \*/

void print\_numbers(void)

{

 int c;

 for (c = 48; c < 58; c++)

 {

 \_putchar(c);

 }

 \_putchar('\n');

}

==============================================

4-print\_most\_numbers.c CODE

#include "main.h"

/\*\*

 \* print\_most\_numbers - checks for checks for a digit (0 through 9).

 \*

 \* Return: Always 0.

 \*/

void print\_most\_numbers(void)

{

 int c;

 for (c = 48; c < 58; c++)

 {

 if (c != 50)

 {

 if (c != 52)

 {

 \_putchar(c);

 }

 }

 }

 \_putchar('\n');

}

============================================

5-more\_numbers.c CODE

#include "main.h"

/\*\*

 \* more\_numbers - prints 10 times the numbers, from 0 to 14

 \* followed by a new line

 \*/

void more\_numbers(void)

{

 int i, j;

 for (i = 0; i < 10; i++)

 {

 for (j = 0; j < 15; j++)

 {

 if (j >= 10)

 \_putchar(j / 10 + '0');

 \_putchar(j % 10 + '0');

 }

 \_putchar('\n');

 }

}

================================================

6-print\_line.c CODE

#include "main.h"

/\*\*

 \* print\_line - draws a straight line in the terminal

 \* @n: number of times the character \_ should be printed

 \*/

void print\_line(int n)

{

 if (n <= 0)

 {

 \_putchar('\n');

 } else

 {

 int i;

 for (i = 1; i <= n; i++)

 {

 \_putchar('\_');

 }

 \_putchar('\n');

 }

}

==============================================

7-print\_diagonal.c CODE

#include "main.h"

/\*\*

 \* print\_diagonal - draws a diagonal line on the terminal

 \* @n: number of times the character \ should be printed

 \*/

void print\_diagonal(int n)

{

 if (n <= 0)

 {

 \_putchar('\n');

 } else

 {

 int i, j;

 for (i = 0; i < n; i++)

 {

 for (j = 0; j < n; j++)

 {

 if (j == i)

 \_putchar('\\');

 else if (j < i)

 \_putchar(' ');

 }

 \_putchar('\n');

 }

 }

}

=============================================

8-print\_square.c CODE

#include "main.h"

/\*\*

 \* print\_square - prints a square, followed by a new line;

 \* @size: size of the square

 \*/

void print\_square(int size)

{

 if (size <= 0)

 {

 \_putchar('\n');

 } else

 {

 int i, j;

 for (i = 0; i < size; i++)

 {

 for (j = 0; j < size; j++)

 {

 \_putchar('#');

 }

 \_putchar('\n');

 }

 }

}

===============================================

9-fizz\_buzz.c CODE

#include "main.h"

#include <stdio.h>

/\*\*

 \* main - prints the numbers from 1 to 100, followed by a new line

 \* but for multiples of three prints Fizz instead of the number

 \* and for the multiples of five prints Buzz

 \* Return: Always 0 (Success)

 \*/

int main(void)

{

 int i;

 for (i = 1; i <= 100; i++)

 {

 if (i % 3 == 0 && i % 5 != 0)

 {

 printf(" Fizz");

 } else if (i % 5 == 0 && i % 3 != 0)

 {

 printf(" Buzz");

 } else if (i % 3 == 0 && i % 5 == 0)

 {

 printf(" FizzBuzz");

 } else if (i == 1)

 {

 printf("%d", i);

 } else

 {

 printf(" %d", i);

 }

 }

 printf("\n");

 return (0);

}

==============================================

10-print\_triangle.c CODE

#include "main.h"

/\*\*

 \* print\_triangle - prints a triangle, followed by a new line

 \* @size: size of the triangle

 \*/

void print\_triangle(int size)

{

 if (size <= 0)

 {

 \_putchar('\n');

 } else

 {

 int i, j;

 for (i = 1; i <= size; i++)

 {

 for (j = i; j < size; j++)

 {

 \_putchar(' ');

 }

 for (j = 1; j <= i; j++)

 {

 \_putchar('#');

 }

 \_putchar('\n');

 }

 }

}

===========================================

100-prime\_factor.c CODE

#include <stdio.h>

#include <math.h>

/\*\*

 \* main - finds and prints the largest prime factor of the number 612852475143

 \* followed by a new line

 \* Return: Always 0 (Success)

 \*/

int main(void)

{

 long int n;

 long int max;

 long int i;

 n = 612852475143;

 max = -1;

 while (n % 2 == 0)

 {

 max = 2;

 n /= 2;

 }

 for (i = 3; i <= sqrt(n); i = i + 2)

 {

 while (n % i == 0)

 {

 max = i;

 n = n / i;

 }

 }

 if (n > 2)

 max = n;

 printf("%ld\n", max);

 return (0);

}

=============================================

101-print\_number.c CODE

#include "main.h"

/\*\*

 \* print\_number - prints an integer

 \* @n: integer to be printed

 \*/

void print\_number(int n)

{

 unsigned int n1;

 if (n < 0)

 {

 n1 = -n;

 \_putchar('-');

 } else

 {

 n1 = n;

 }

 if (n1 / 10)

 {

 print\_number(n1 / 10);

 }

 \_putchar((n1 % 10) + '0');

}

#include "main.h"

/\*\*

\* main - tests function that prints if integer is positive or negative

\* Return: 0

\*/

int main(void)

{

int i;

i = 0;

positive\_or\_negative(i);

return (0);

}

#include <stdio.h>

/\*\*

\* main - causes an infinite loop

\* Return: 0

\*/

int main(void)

{

int i;

printf("Infinite loop incoming :(\n");

i = 0;

/\*

\* Commented out to avoid infinite loop

\* while (i < 10)

\* {

\* putchar(i);

\* }

\*/

printf("Infinite loop avoided! \\o/\n");

return (0);

}

Gggggg

#include "main.h"

/\*\*

\* largest\_number - returns the largest of 3 numbers

\* @a: first integer

\* @b: second integer

\* @c: third integer

\* Return: largest number

\*/

int largest\_number(int a, int b, int c)

{

int largest;

if (a >= b && a >= c)

{

largest = a;

}

else if (b >= a && b > c)

{

largest = b;

}

else

{

largest = c;

}#include "main.h"

/\*\*

\* print\_number - prints an integer

\* @n: integer to be printed

\*/

void print\_number(int n)

{

unsigned int n1;

if (n < 0)

{

n1 = -n;

\_putchar('-');

}

else

{

n1 = n;

}

if (n1 / 10)

{

print\_number(n1 / 10);

}

\_putchar((n1 % 10) + '0');

}

return (largest);

}

#include "main.h"

/\*\*

\* print\_number - prints an integer

\* @n: integer to be printed

\*/

void print\_number(int n)

{

unsigned int n1;

if (n < 0)

{

n1 = -n;

\_putchar('-');

}

else

{

n1 = n;

}

if (n1 / 10)

{

print\_number(n1 / 10);

}

\_putchar((n1 % 10) + '0');

}

#include <stdio.h>

#include "main.h"

/\*\*

\* print\_remaining\_days - takes a date and prints how many days are

\* left in the year, taking leap years into account

\* @month: month in number format

\* @day: day of month

\* @year: year

\* Return: void

\*/

void print\_remaining\_days(int month, int day, int year)

{

if ((year % 4 == 0 ) && ( year % 400 == 0 || year % 100 != 0))

{

if (month >= 2 && day >= 60)

{

day++;

}

printf("Day of the year: %d\n", day);

printf("Remaining days: %d\n", 366 - day);

}

else

{

if (month == 2 && day == 60)

{

printf("Invalid date: %02d/%02d/%04d\n", month, day - 31, year);

}

else

{

printf("Day of the year: %d\n", day);

printf("Remaining days: %d\n", 365 -

#!/bin/bash

Change owner and group owner of \_hello to vincent and staff

sudo chown vincent:staff \_hello

#include "main.h"

#include <stdio.h>

/\*\*

\* print\_times\_table - prints the n times table, starting with 0

\* @n: the number for which the times table is printed

\*/

void print\_times\_table(int n)

{

if (n >= 0 && n <= 15)

{

int i, j, result;

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

{

for (j = 0; j <= n; j++)

{

result = i \* j;

if (j == 0)

printf("%d", result);

else

printf(",%3d", result);

}

printf("\n");

}

}

}

If I were an App, I would be Bing Maps.

Bing Maps is a web mapping service that provides various features such as route planning, traffic information, street view, 3D maps, and more. It works by using satellite imagery, aerial photography, and vector data to create interactive maps that users can explore and navigate. Bing Maps is a creative and innovative App for the digital era, as it offers many advantages and best practices that can help users with various tasks and needs.

One of the advantages of Bing Maps is that it can help users find the best route to their destination, whether they are driving, walking, cycling, or using public transportation. Bing Maps can calculate the distance, time, and cost of different routes, and suggest the optimal one based on the user's preferences and current conditions. Bing Maps can also provide real-time traffic information, such as congestion, accidents, road closures, and construction, and update the route accordingly. Bing Maps can also show alternative routes and modes of transportation, and compare them side by side, to give users more options and flexibility.

Another advantage of Bing Maps is that it can help users discover new places, such as restaurants, hotels, shops, landmarks, and attractions. Bing Maps can show the ratings, reviews, photos, and details of various places, and allow users to filter and sort them by different criteria, such as distance, price, category, and popularity. Bing Maps can also show the opening hours, contact information, and directions of the places, and enable users to book, reserve, or order them online. Bing Maps can also integrate with social media platforms, such as Facebook, Twitter, and Instagram, and allow users to share their location, check in, and post reviews and photos of the places they visit.

Another advantage of Bing Maps is that it can provide a realistic and immersive view of the world, using street view, 3D maps, and augmented reality. Bing Maps can show the street-level imagery of various locations, and allow users to pan, zoom, and rotate the view as if they were there. Bing Maps can also show 3D models of buildings, landmarks, and terrain, and allow users to fly over and around them as if they were in a helicopter. Bing Maps can also use augmented reality to overlay digital information and graphics on the real world, and allow users to interact with them as if they were in a video game.

Bing Maps is not only a creative and innovative App, but also a versatile and adaptable one, as it can integrate with other Apps and services, such as Cortana, Outlook, Skype, and Microsoft 365, to provide a seamless and convenient experience for users. For example, Bing Maps can work with Cortana, a virtual assistant, to provide voice commands and natural language processing, and allow users to ask questions and get answers about the maps and places. Bing Maps can also work with Outlook, an email and calendar service, to show the location and directions of the events and appointments, and allow users to add them to their itinerary. Bing Maps can also work with Skype, a video and voice calling service, to show the location and time zone of the contacts, and allow users to call them directly from the maps. Bing Maps can also work with Microsoft 365, a cloud-based productivity suite, to show the location and data of documents, spreadsheets, and presentations, and allow users to edit and share them online.

However, Bing Maps is not without its challenges and limitations, as it may face some issues with data quality, coverage, and consumption. One of the challenges of Bing Maps is that it may not have the most accurate or up-to-date information for some regions or locations, especially in remote or rural areas, where satellite imagery, aerial photography, and vector data may be sparse, outdated, or incomplete. This may affect the reliability and usefulness of the maps and features, and cause errors or inaccuracies in the route planning, traffic information, and place discovery. Another challenge of Bing Maps is that it may consume a lot of data and battery power when used on mobile devices, especially when using street view, 3D maps, and augmented reality features, which require high-resolution images and graphics, and intensive processing and rendering. This may affect the performance and efficiency of the devices, and cause slow loading, lagging, or crashing of the App.

To overcome these challenges and limitations, Bing Maps can adopt some strategies and solutions, such as improving the data quality and coverage and optimizing the data consumption and battery usage. One of the strategies to improve the data quality and coverage of Bing Maps is to collaborate with local authorities, businesses, and communities, and use their data sources and expertise to update and enrich the maps and features. For example, Bing Maps can partner with local governments, transportation agencies, and tourism boards, to get the latest and official information on the roads, traffic, and places. Bing Maps can also partner with local businesses, such as restaurants, hotels, and shops, to get the latest and verified information on their ratings, reviews, and details. Bing Maps can also partner with local communities, such as volunteers, travelers, and enthusiasts, to get the latest and diverse information on street view, 3D maps, and augmented reality. Bing Maps can also use user feedback and crowdsourcing to collect and validate the information from the users and reward them with incentives, such as points, badges, and discounts.

Another strategy to optimize the data consumption and battery usage of Bing Maps is to use compression, caching, and offline modes, and reduce the amount and frequency of the data transfer and processing. For example, Bing Maps can use compression techniques, such as lossy and lossless compression, to reduce the size and quality of the images and graphics, and make them load faster and smoother. Bing Maps can also use caching techniques, such as local and cloud caching, to store the images and graphics on the device or the server, and make them load from there, instead of downloading them again. Bing Maps can also use offline modes, such as pre-downloading and saving the maps and features, and make them available without an internet connection.

In conclusion, Bing Maps is a creative and innovative App for the digital era, as it provides various features and benefits that can help users with various tasks and needs, such as route planning, traffic information, street view, 3D maps, and more. Bing Maps is also a versatile and adaptable App, as it can integrate with other Apps and services, such as Cortana, Outlook, Skype, and Microsoft 365, to provide a seamless and convenient experience for users. However, Bing Maps also faces some challenges and limitations, such as data quality, coverage, and consumption, which may affect its reliability and usefulness. To address these issues, Bing Maps can adopt some strategies and solutions, such as collaborating with local authorities, businesses, and communities, using user feedback and crowdsourcing, to improve the data quality and coverage, and using compression, caching, and offline modes, to optimize the data consumption and battery usage. Bing Maps is a great example of a creative and innovative App that can enhance the top 50 Apps ever made, and help the Developer Committee Leaders shape their improvement plan for the next decade.