# Data Structures And Algorithms 2 Project

Farid Muradzada and Akif Mursalov
UFAZ, CS-20

May 15, 2023

## Contents

## 1 Abstract

The symasym application is a code that utilizes randomization to create 32x32.bmp images featuring either vertical or horizontal lines. The lines have a thickness of three pixels and can exhibit symmetric or asymmetric shapes. The generation of images is based on a provided seed value, which determines the starting locations and movement patterns of the lines.

For vertical lines, those starting at the bottom of the image descend with a variable x-value, while lines starting at the top move from left to right with a variable y-value. The application generates image files with filenames beginning with "h" or "v" to indicate the direction of the lines, followed by "s" or "a" to denote symmetric or asymmetric shapes, respectively. Additionally, the seed number is included in the filename to ensure reproducibility.

By combining randomization, line thickness, symmetry, and variable movement patterns, the symasym application offers a versatile tool for generating diverse 32x32.bmp images of lines, enabling users to explore and manipulate visual patterns within this framework.

## 2 Introduction

The program is designed to create .bmp images featuring vertical or horizontal lines, with the option to generate symmetric or asymmetric shapes. The program operates based on various command-line options and utilizes a seed value for randomization.
When executed without any arguments, the program generates a default 32x32 white image with a black 1-pixel frame. The image contains a continuous line with a thickness of 3 pixels, which starts either vertically or horizontally depending on the specified side. The starting point and shape of the line are determined randomly based on the seed value.
For vertical lines, if the starting point is on the left side of the image, the line progresses from left to right with a variable y-value, allowing it to randomly move up or down. The line remains continuous even when it reaches positions 3 pixels above or below the current position.
In the case of horizontal lines, if the starting point is on the top side of the image, the line progresses from top to bottom with a variable x-value, allowing it to randomly move left or right. The line remains continuous even when it reaches positions 3 pixels to the left or right of the current position.
The program offers several command-line options to modify its behavior, such as specifying the line's symmetry, orientation, seed number, output filename, and image size. These options provide flexibility in creating multiple images with varying characteristics.

## 3 Code implementation

### 3.1 Libraries

The code utilizes several libraries that provide essential functionalities. These libraries include:

- **<stdio.h>**: This library allows input and output operations, enabling functions like printf for printing output and fopen for file handling.
- **<stdlib.h>**: This library offers general-purpose functions such as memory allocation (**malloc**) and random number generation (**rand**, **srand**).
- **<stdint.h>**: This library defines integer types with specific widths, such as uint16_t and uint32_t, ensuring consistent sizes across different platforms.
- **<string.h>**: This library provides functions for string manipulation, such as strcmp for string comparison and memset for setting memory blocks.
- **<time.h>**: This library includes functions for working with time and date. In the code, it is used to set the seed for random number generation based on the current time (**time(NULL)**).

By including these libraries in your code, you gain access to their respective functions, types, and macros, enabling you to perform various operations efficiently.

### 3.2 Functions

**void fillImage(unsigned char *imageData, int width, int height, int stride, int bytesPerPixel, int startY, int thickness)**: This function fills a portion of an image with black color. It takes the image data, image dimensions (width and height), stride (number of bytes per row), bytes per pixel, starting y-coordinate, and thickness as input. It then iterates over the pixels in the specified portion and sets the RGB values to 0 (black).

**void generateImage(int size, int thickness, Direction direction, const char *filename, int *points)**:This function generates an image with a polyline pattern. It takes the size of the image, thickness of the polyline, direction (vertical or horizontal), filename for the output image, and an array of points as input. The function creates an image buffer, fills it with white color, and then draws the polyline on the image by setting the corresponding pixels to black. Finally, it creates a BMP header, writes the image data to a file, and saves it as a BMP image.

**int *generatePoints(int size, Symmetry symmetry, int thickness)**: This function generates an array of points for drawing a polyline. It takes the size of the image, symmetry type (symmetric or asymmetric), and thickness of the polyline as input. The function dynamically allocates memory for the points array, randomly generates the starting point, and calculates the subsequent points by adding or subtracting a random value within the thickness range. If the symmetry is set to symmetric, the function ensures that the points on one half of the image are mirrored on the other half. The function returns the array of points.

These special functions play crucial roles in the generation and manipulation of images in the code. They encapsulate specific tasks and provide the necessary functionality to create polyline patterns and save them as BMP images. Understanding these functions is essential for comprehending the overall functionality of the code.

### 3.3 LAUNCHING THE CODE

To run the code with the provided CMakeLists.txt file, you can follow these steps:

1. Create a new directory for building the project (referred to as the "build directory"). This directory should be separate from the directory containing the source code files.

2. Open a terminal or command prompt and navigate to the build directory.

3. Run the CMake command followed by the path to the directory containing the CMakeLists.txt file. In this case, the command would be:

   **cmake /path/to/CMakeLists.txt**

   Replace *"/path/to/CMakeLists.txt"* with the actual path to the directory containing the CMakeLists.txt file.

4. After running the CMake command, it will generate the build files according to the instructions in the CMakeLists.txt file. The specific build files depend on the platform and generator used.

5. Once the build files are generated, you can build the project by running the appropriate build command for your platform. For example:

   On Unix/Linux systems: Run make.After the build process completes successfully, you will find the executable file (named "symasym" in this case) in the build directory.

6. Run the executable by executing its name in the terminal or command prompt: *./symasym*
   Note that the exact command may vary depending on your operating system.

   By following these steps, you should be able to build and run the code using the provided CMakeLists.txt file.

## 4 Conclusion

In conclusion, the implementation of the BMP image creation project in C was a success, as it achieved its objective of generating a BMP image from the ground up. The project involved a thorough understanding of the BMP file format, the development of a function to construct the BMP header, and the writing of pixel data to the file. Throughout the project, various challenges were faced and overcome, including the need to ensure correct alignment of pixel data. Overall, this project served as an invaluable learning opportunity, allowing for the acquisition of essential skills in file manipulation, data handling, and working with binary data in the C programming language.