Lab 1 Report

Robotics Integration Group Project I

Yuwei ZHAO (23020036096)
Group #31 2025-11-06

Abstract

This report presents the setup and results of Lab 1. The main objective of the experiment is to install Ubuntu 20.04, Git, and the C++ compilation tools, and to become familiar with basic shell commands and version control using Git.

See Resources on git@github.com:RamessesN/Robotics_MIT.

1 Introduction

This laboratory session serves as the foundation for subsequent robotics development projects. Before implementing control algorithms or integrating robotic systems, it is essential to establish a consistent and reliable development environment. The primary purpose of this experiment is to install and configure Ubuntu 20.04, Git, and C++ compilation tools, ensuring that all essential development utilities are functional.

2 Procedure

2.1 Part I

2.1.1 Objective

To set up an Ubuntu environment on a virtual machine for subsequent experiments.

2.1.2 Methodology

The experiment was conducted on an *Apple Silicon* Mac using *Parallels Desktop* as the virtualization platform. Since Ubuntu 20.04 does not provide an official ARM64 image, the ARM version of Ubuntu 24.04 was selected instead. The downloaded ISO file was mounted as the installation source, and the operating system was successfully installed by following the guided setup procedure.

Note: Compatibility testing confirmed that ROS1 and related tools can be properly installed and executed on Ubuntu 24.04.

2.1.3 Observations

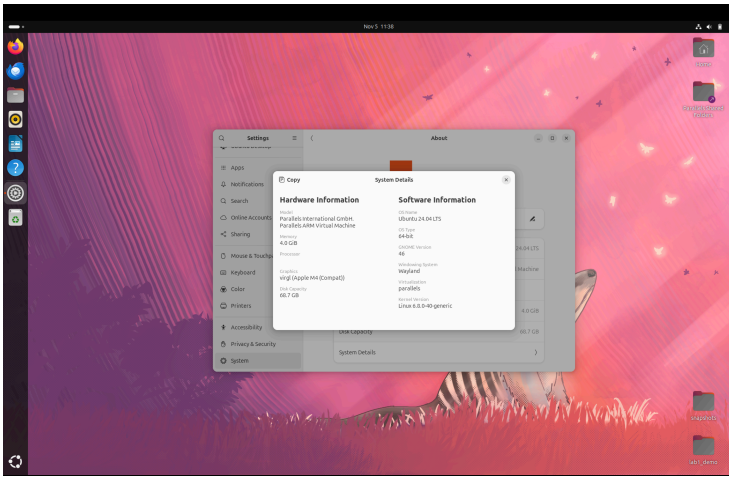


Figure 1: Ubuntu 24.04 Installed on Parallels Desktop

2.1.4 Discussion

The installation of Ubuntu 24.04 on Parallels Desktop was completed successfully, providing a stable ARM-based Linux environment for further development.

2.2 Part II

2.2.1 Objective

To install essential development tools required for subsequent software compilation and version control, including `Git` and `build-essential`.

2.2.2 Methodology

The following command was executed in the terminal to install the necessary packages:

```
sudo apt install build-essential cmake git
```

This command installs the GNU compiler collection, build utilities, CMake, and Git.

2.2.3 Observations

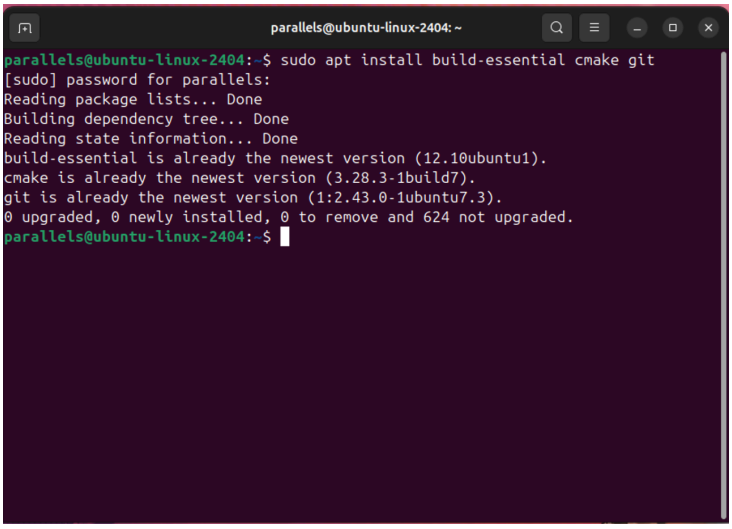


Figure 2: Installation of Essential Packages

2.2.4 Discussion

The packages were successfully installed, providing a complete C/C++ build environment and version control support. These tools form the foundation for compiling robotics-related source code and managing project repositories in subsequent experiments.

2.3 Part III

2.3.1 Objective

To get familiar with the basic usage of Git and the process of compiling a simple C++ project using `CMake`.

2.3.2 Methodology

- **Cloning the Repository:**

  ```
  git clone git@github.com:RamessesN/VesselContest_F1.git
  ```

  Note: This repository is one of my personal projects on GitHub and is used here only for testing Git operations.

- **Project Structure:** The project folder was organized as follows:

  ```
  lab1_demo/
  - CMakeLists.txt
  - main.cpp
  ```

- my_lib.hpp
- my_lib.cpp

- **CMake Configuration File (`CMakeLists.txt`):**

```
cmake_minimum_required(VERSION 3.10)

project(hello_world VERSION 1.0 LANGUAGES CXX)

add_library(mylib my_lib.cpp my_lib.hpp)
add_executable(main main.cpp)
target_link_libraries(main PRIVATE mylib)
```

- **Source Code Files:**
  main.cpp

```cpp
#include <iostream>
#include "my_lib.hpp"

int main() {
    std::cout << "Hello world!" << std::endl;
    std::cout << my_lib_function() << std::endl;
    return 0;
}
```

  my_lib.cpp

```cpp
#include <string>
#include "my_lib.hpp"

std::string my_lib_function() {
    return "In library";
}
```

  my_lib.hpp

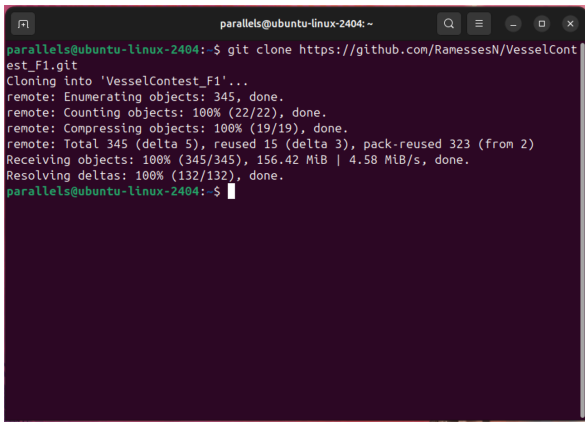
```cpp
#pragma once
#include <string>

std::string my_lib_function();
```

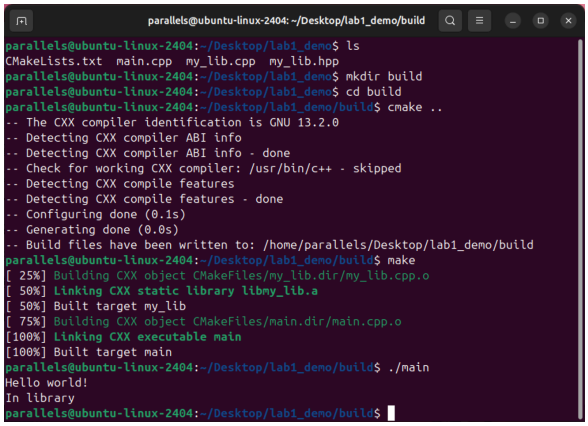
- **Compilation Commands:** The following commands were used to configure and build the project:

```
mkdir build && cd build
cmake ..
make
./main
```

2.3.3 Observations



(a) Git clone command execution



(b) Compilation process

Figure 3: Git cloning and project compilation

2.3.4 Discussion

The Git repository was successfully cloned from GitHub and the C++ project was compiled without any errors.

2.4 Part IV

2.4.1 Objective

1. **Git**
   Clone the repository github@ouc-vlab-course/vnav-codes to a folder

2. **Shell**
   Download file @dante.txt (try using $wget$ in bash, if network not accessable, try add https://ghp.ci/ before the file link), then finish these exercise:

   exercise 1(use command wc): create file $exercise1.txt$, calculate thses information: (1) this file contains how many lines
   (2) this file contains how many words
   (3) how many lines are not blank

   exercise 2: using apt to install fortune-mod, Run $fortune$ 5 more times and each time redirect the output to a file called $fortunes.txt$ in `~/vnav-personal/lab1` (Hint: do not recreate the file 5 times - each time a new proverb should be added to the end of $fortunes.txt$)

3. **C++**

   - Operators

     (a) What are the values of i and j after running the following code.

```
int i = 0, j;
j = ++i;
j = i++;
```

(b) What does the following code print?

```
int i = 42;
std::string output = (i < 42) ? "a" : "b";
std::cout << output << std::endl;
```

- References and Pointers

(a) What does the following code print?

```
int i;
int& ri = i;
i = 5;
ri = 10;
std::cout << i << " " << ri << std::endl;
```

(b) What does the following code print?

```
int i = 42;
int* j = &i;
*j = *j**j;
std::cout << *j << std::endl;
```

(c) What does the following code print?

```
int i[4] = {42, 24, 42, 24};
*(i + 2) = *(i + 1) - i[3];
std::cout << *(i + 2) << std::endl;
```

(d) What does the following code print?

```
void reset(int &i) {
    i = 0;
}

int j = 42;
reset(j);
std::cout << j << std::endl;
```

- Numbers

(a) What are the differences between int, $long$, $longlong$, and $short$?

(b) What are the differences between a $float$ and $double$? What is the value of i after running the following code snippet?

```
int i;
i = 3.14;
```

(c) What are the differences between an unsigned and signed type? What is the value of c in the following code snippet assuming $chars$ are 8-bit?

```
unsigned char c = -1;
```

(d) What will the value of i be after running the following code snippet?

```
int i = 42;
if (i) {
 i = 0;
```

```
} else {
 i = 43;
}
```

4. **C++ Extension**
   Create a folder called $RandomVector$, and put the code of lab1 to this folder.

   The class $RandomVector$ defined in the header file $random_vector.h$ abstract a vector of doubles. The following methods are required to implement:

   - $RandomVector(intsize, doublemax_val = 1)$ (constructor): initialize a vector of doubles of size $size$ with random values between 0 and max_val (default value 1)
   - $doublemean()$ returns the mean of the values in random vector
   - $doublemax()$ returns the max of the values in random vector
   - $doublemin()$ returns the min of the values in random vector
   - $voidprint()$ prints all the values in the random vector
   - $voidprintHistogram(intbins)$ computes the histogram of the values using $bins$ number of bins between $min()$ and $max()$ and print the histogram itself (see the example below).

   To do so complete all the TODOs in the file $random_vector.cpp$. When you are done compile the application by running

```
g++ -std=c++11 -Wall -pedantic \
    -o random_vector \
    main.cpp random_vector.cpp
```

   Note: Do not use the function from the $< algorithm >$ header is expected. If complete correctly something of the exercise should be like:

```
$ ./random_vector
0.458724 0.779985 0.212415 0.0667949 0.622538 0.999018 0.489585 \
0.460587 0.0795612 0.185496 0.629162 0.328032 0.242169 0.139671 \
0.453804 0.083038 0.619352 0.454482 0.477426 0.0904966
Mean: 0.393617
Min: 0.0667949
Max: 0.999018
Histogram:
***     ***
***     ***
***     ***
***     ***
***     ***
***     ***
***     *** ***
*** *** *** *** ***
```

2.4.2 Methodology

- **Git**
  The repository `vnav-codes` was cloned using the command:

  ```
  git clone https://github.com/ouc-vlab-course/vnav-codes
  ```

- **Shell**
  The file `dante.txt` was downloaded using the command:

  ```
  wget \
  https://raw.githubusercontent.com/dlang/dmd/master/druntime/ \
  benchmark/extra-files/dante.txt
  ```

  For exercise1: run the following commands to check the $lines$, $words$ and $bytes$ of the $dante.txt$:

  ```
  wc dante.txt >> exercise1.txt
  ```

  To count the number of non-blank lines, run:

  ```
  echo "Non-blank lines: $(grep -v '^$' dante.txt | wc -l)" \
  >> exercise1.txt
  ```

  Therefore, all the required information can be got from the above commands:

  - `echo "Total lines: $(wc -l < dante.txt)" > exercise1.txt`
  - `echo "Total words: $(wc -w < dante.txt)" >> exercise1.txt`
  - `echo "Non-blank lines: $(grep -v '^$' dante.txt | wc -l)" \`
    `>> exercise1.txt`

  Then, use $more$ command to check the content of `exercise1.txt`:

  ```
  more exercise1.txt
  ```

  For exercise2: run the following command to get $fortune - mod$ installed:

  ```
  sudo apt install fortune-mod
  ```

  Then, run the command below 5 times to get a random quote and use $more$ to check:

  ```
  fortune-mod >> fortunes.txt
  more fortunes.txt
  ```

- **C++**

– **Operators**

1. i = 2, j = 1
   At first, i equals to 0. After executing `j = ++i;`, i is incremented to 1 and then assigned to j. After executing `j = i++;`, j is assigned the current value of i(which is 1), and then i is incremented to 2.
2. b
   i = 42 is not less than 42, so the condition `(i < 42)` evaluates to false. Therefore, the expression evaluates to ”b”.

– **References and Pointers**

1. i = 10, ri = 10
   At first, i is declared but not initialized. Then, ri is defined as a reference(&) to i. When `i = 5;` is executed, i is assigned the value 5. When `ri = 10;` is executed, it modifies the value of i through the reference ri, setting i to 10. Thus, both i and ri will print 10.
2. 1764
   Initially, i is set to 42, and j is a pointer() that points to i. The expression `*j = (*j) * (*j);` dereferences j to get the value of i (which is 42), squares it (42 * 42 = 1764). Then assign the result back to the location pointed to by j.
3. 0
   An array i of 4 integers is initialized with the values 42, 24, 42, 24. The expression `*(i + 2) = *(i + 1) - i[3];` calculates the value at index 1 (which is 24) minus the value at index 3 (which is also 24), resulting in 0. This value is then assigned to the location at index 2 of the array.
4. 0
   The function `reset` takes an integer reference(&) as its parameter and sets the value of that integer to 0. When `reset(j);` is called, it modifies the value of j to 0.

– **Numbers**

1. Comparison of Integer Types

Type	Size (bytes)	Range
int	4	-2,147,483,648 to 2,147,483,647
long	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
long long	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
short	2	-32,768 to 32,767

Table 1: Comparison of Integer Types

2. Comparison of Floating-Point Types

Type	Size (bytes)	Precision
float	4	about 7 decimal digits
double	8	about 15 decimal digits

Table 2: Comparison of Floating-Point Types

i = 3.
After executing `i = 3.14;`, the floating-point value 3.14 is truncated to 3 when assigned to the integer variable i.

3. Signed & Unsigned Types
Signed types can represent both positive and negative values, while unsigned types can only represent non-negative values.

c = 255.
Assigning -1 to an unsigned char will result in wrap-around behavior. Since unsigned char can only hold values from 0 to 255, -1 wraps around to 255. Therefore, the value of c will be 255.

4. i = 0
The condition in the if statement checks whether i is non-zero. Since i is initially 42 != 0, the condition evaluates to true, and the code inside the if block is executed, setting i to 0.

- **C++ Extension**

1. Created a folder named *RandomVector* using the command:

```
mkdir RandomVector
```

Put the code in lab1 into this folder using the command:

```
mv lab1/* RandomVector/
```

2. Implemented the required methods in the `random_vector.cpp` file:
   - `RandomVector(int size, double max_val = 1)`:
   In order to utilize $srand(314159)$; in *main.cpp*, I decided to call `std::rand()` directly without reseeding it in the constructor. The random values are generated using the formula:

```
static_cast<double>(rand()) / RAND_MAX * max_val
```

   This formula generates a random double value between 0 and `max_val`.
   - `void print()`:
   Iterated through the vector and printed each value followed by a space. Besides, every 9 values, a newline character is printed to enhance readability.

```
for(size_t i = 0; i < vect.size(); i++) {
    printf("%.6lf ", vect[i]);
    if((i + 1) % 9 == 0)
      cout << endl;
  }
```

- `double mean()`:
  Calculated the sum of all elements in the vector and divided it by the size of the vector to obtain the mean value.

```
for(size_t i = 0; i < vect.size(); i++)
    sum += vect[i];
return sum / vect.size();
```

- `double max()`:
  Found the maximum element in the vector by iterating through it.

```
if(vect[i] > max_val)
    max_val = vect[i];
```

- `double min()`:
  Found the minimum element in the vector by iterating through it.

```
if(vect[i] < min_val)
    min_val = vect[i];
```

- `void printHistogram(int bins)`:
  Computed the histogram by first determining the range of values (min to max) and then calculating the width of each bin. Counted the number of elements that fall into each bin and stored the counts in a vector. Finally, printed the histogram by scaling the counts to fit within the maximum count.

```
double bin_size = (max_val - min_val) / bins;
vector<int> counts(bins, 0);
for(double val : vect) {
    int index = static_cast<int>((val - min_val) / bin_size);
    if(index == bins) index--;
    counts[index]++;
}
int max_count = 0;
  for(int count : counts)
    if(count > max_count) max_count = count;

  for(int level = max_count; level > 0; level--) {
    for(int bin = 0; bin < bins; bin++) {
      if(counts[bin] >= level)
        cout << "***";
      else
        cout << "   ";
      cout << " ";
    }
    cout << endl;
  }
```

3. Built the project using the command:

```
g++ -std=c++11 -Wall -pedantic -o \
random_vector main.cpp random_vector.cpp
```

4. Executed the program using the command: `./random_vector`
   Then, the result was obtained as expected:

```
0.458724 0.779985 0.212415 0.0667949 0.622538 0.999018 0.489585 \
0.460587 0.0795612 0.185496 0.629162 0.328032 0.242169 0.139671 \
```

```
0.453804 0.083038 0.619352 0.454482 0.477426 0.0904966
Mean: 0.393617
Min: 0.0667949
Max: 0.999018
Histogram:
***      ***
***      ***
***      ***
***      ***
***      ***
***      ***
***      *** ***
*** *** *** *** ***
```

Note: The complete code can be found at the end of this report.

2.4.3 Observations

- **Git**

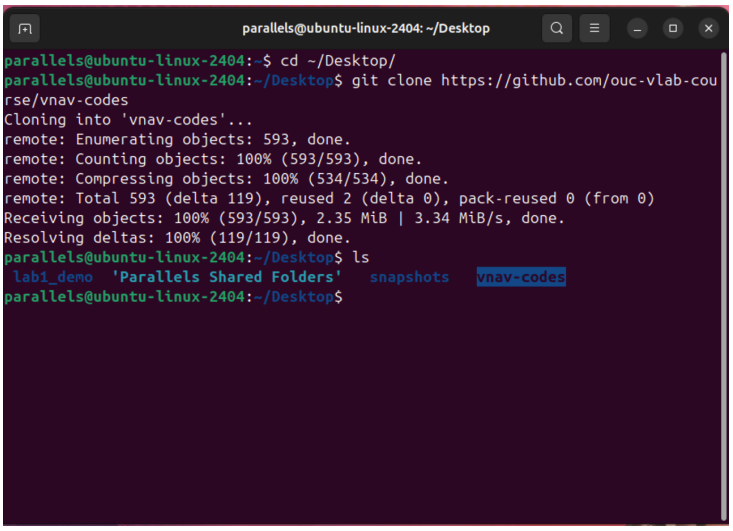


Figure 4: Git Clone of vnav-codes Repository

- **Shell**

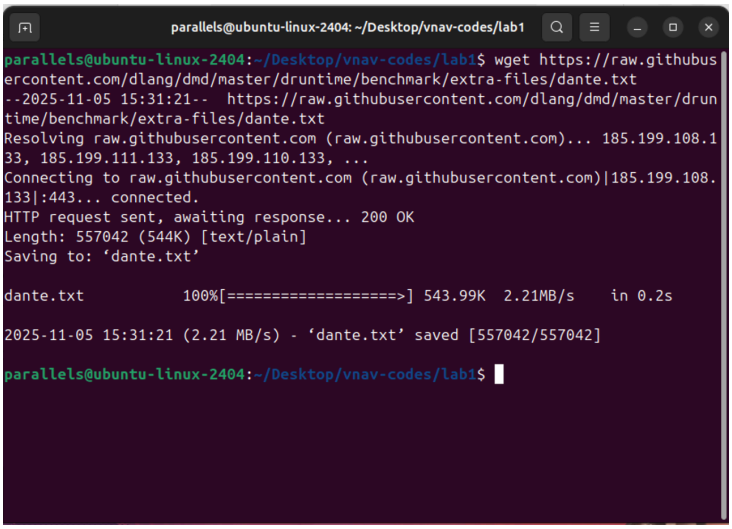


Figure 5: Wget Download of dante.txt

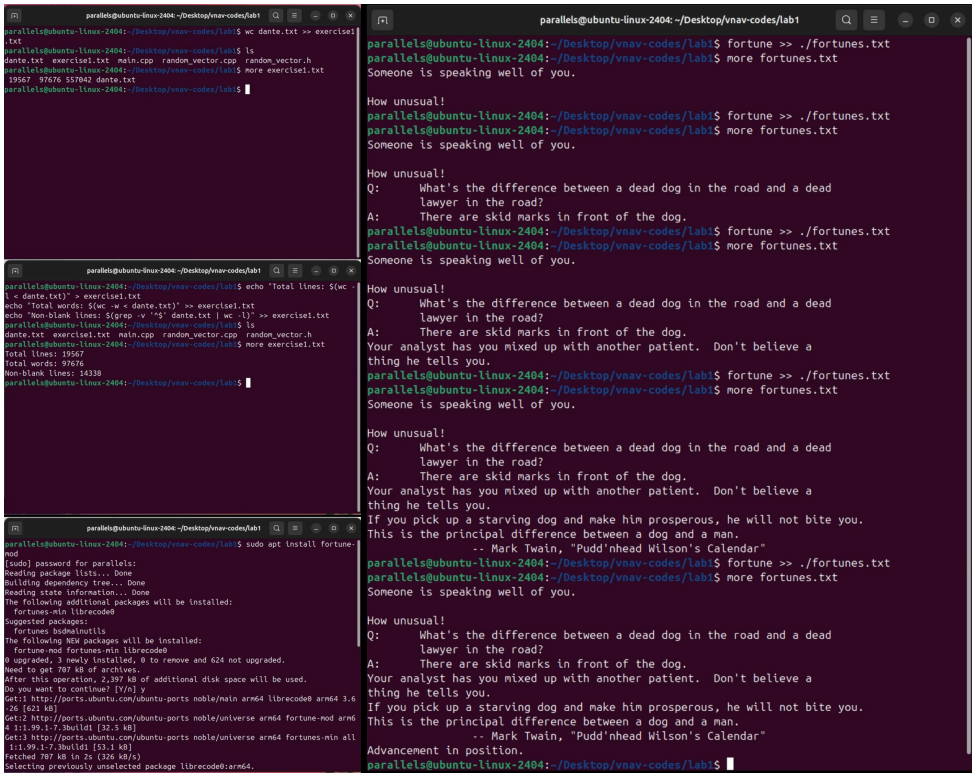


Figure 6: Shell Exercises

- **C++ Extension**

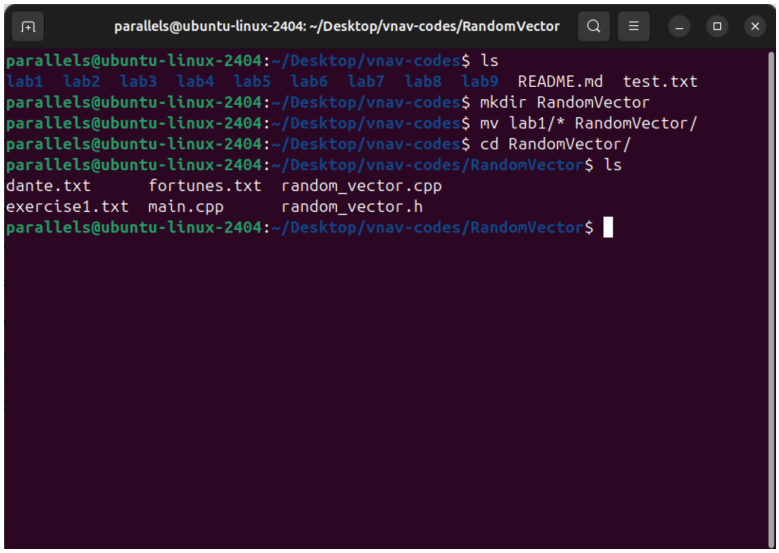


Figure 7: RandomVector Folder Building

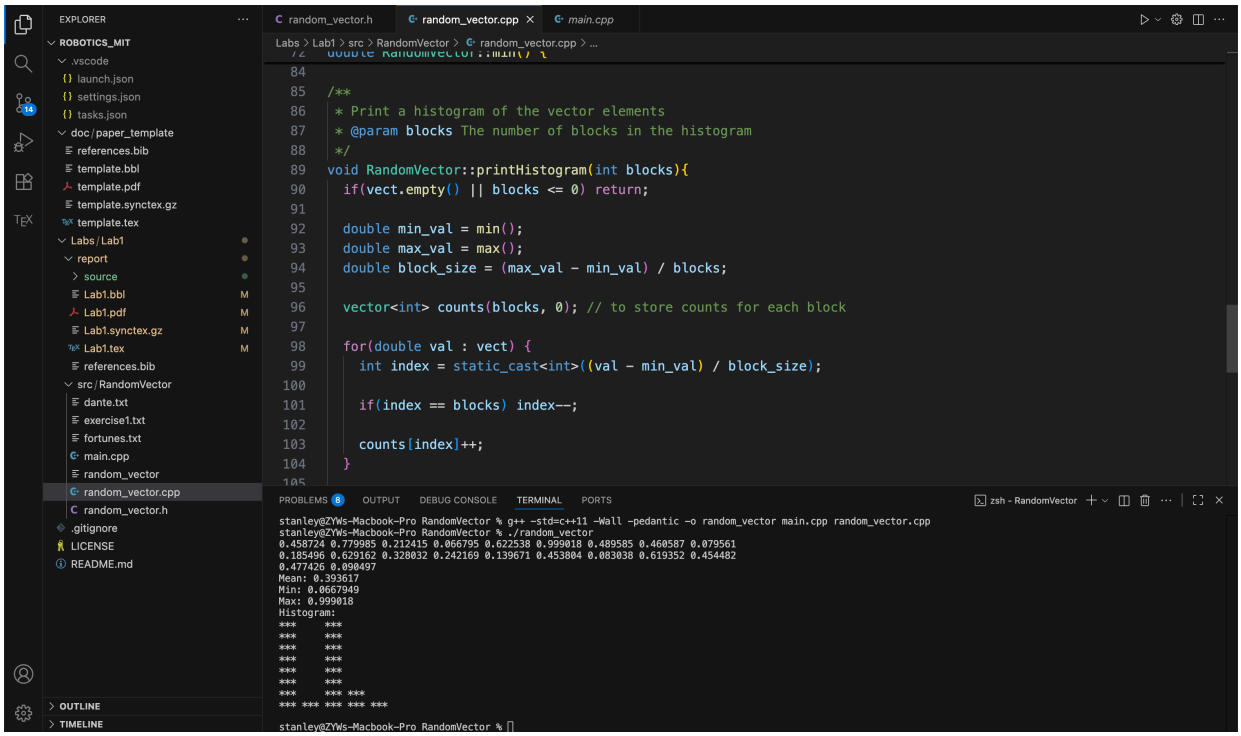


Figure 8: RandomVector Execution Result

2.4.4 Discussion

In this part, I implemented statistical functions and histogram visualization for the *RandomVector* class. To maintain consistency with the random seed already defined in *main.cpp*, I used the traditional C-style random number generator — `srand()` combined with `rand()`.

During the implementation, I avoided using high-level functions from the `<algorithm>` header, such as `max_element()` and `min_element()`. Instead, I manually implemented equivalent functionality using iterative traversal.

The histogram plotting function divides the numerical range between the minimum and maximum values into several equal-width bins and counts the number of elements falling into each bin. This process is conceptually related to the idea of bucket sort — both involve partitioning a range into subintervals and assigning elements to them — but the purpose here is to visualize the frequency distribution of data.

3 Reflection and Analysis

In Part I, I installed a virtual machine environment. Since $AppleSilicon$ Mac does not support traditional $x86$ virtual machine $ISOs$, I selected $ParallelsDesktop$ as the virtualization platform and installed $Ubuntu24.04ARM$. This process allowed me to gain practical experience in setting up a Linux system within a non-native architecture, deepening my understanding of cross-platform development environments.

In Part II, I configured the development environment by installing essential tools such as $build-essential$, $CMake$, and Git using the APT package manager. Through this process, I learned how software package management and development tool configuration work in Linux systems.

In Part III, I cloned a GitHub repository and built a simple C++ project using $CMake$. This part familiarized me with the workflow of using Git for version control, as well as the compilation and linking process in C++ development.

In Part IV, I practiced Git, $Shell$ scripting, and C++ programming through a series of hands-on exercises. These included cloning repositories, processing text files using Shell commands and implementing a C++ class with various functionalities. This part reinforced my understanding of version control and command-line operations while improving my ability to design and implement algorithms in C++.

4 Conclusion

Through this laboratory project, I developed a comprehensive understanding of the software development workflow — from system setup and environment configuration to source code management and program implementation. Each part built upon the previous one, progressively enhancing both my technical proficiency and problem-solving capabilities. By combining Linux command-line tools, Git-based project management, and C++ programming, I not only strengthened my foundational programming skills but also gained valuable experience in practical software engineering workflows.

5 Source Code

- *random_vector.cpp*

```cpp
#include "random_vector.h"
#include <random>
#include <numeric>
#include <iomanip>
#include <cmath>
#include <iostream>

using namespace std;

/**
 * Constructor to initialize the vector with random values
 * @param size The size of the vector
 * @param max_val The maximum value for the random numbers
 */
RandomVector::RandomVector(int size, double max_val) {
    vect.resize(size);
    for(int i = 0; i < size; i++) {
        vect[i] = (double)rand() / RAND_MAX * max_val;
    }
}

/**
 * Print the elements of the vector
 */
void RandomVector::print() {
  for(size_t i = 0; i < vect.size(); i++) {
    printf("%.6lf ", vect[i]);
    if((i + 1) % 9 == 0)
      cout << endl;
  }

  cout << endl;
}


/**
 * Calculate the mean of the vector elements
 * @return The mean value
 */
double RandomVector::mean() {
  if(vect.empty()) return 0.0;

  double sum = 0.0;
  for(size_t i = 0; i < vect.size(); i++) {
    sum += vect[i];
  }

  return sum / vect.size();
}

/**
 * Find the maximum value in the vector
 * @return The maximum value
 */
```

```cpp
double RandomVector::max() {
  if(vect.empty()) return 0.0;

  double max_val = vect[0];
  for(size_t i = 1; i < vect.size(); i++) {
    if(vect[i] > max_val){
      max_val = vect[i];
    }
  }

  return max_val;
}

/**
 * Find the minimum value in the vector
 * @return The minimum value
 */
double RandomVector::min() {
  if(vect.empty()) return 0.0;

  double min_val = vect[0];
  for(size_t i = 1; i < vect.size(); i++) {
    if(vect[i] < min_val){
      min_val = vect[i];
    }
  }

  return min_val;
}

/**
 * Print a histogram of the vector elements
 * @param bins The number of bins in the histogram
 */
void RandomVector::printHistogram(int bins){
  if(vect.empty() || bins <= 0) return;

  double min_val = min();
  double max_val = max();
  double bin_size = (max_val - min_val) / bins;

  vector<int> counts(bins, 0); // to store counts for each bin

  for(double val : vect) {
    int index = static_cast<int>((val - min_val) / bin_size);

    if(index == bins) index--;

    counts[index]++;
  }

  int max_count = 0;
  for(int count : counts)
    if(count > max_count) max_count = count;
    // find the maximum count for scaling

  for(int level = max_count; level > 0; level--) {
    for(int bin = 0; bin < bins; bin++) {
      if(counts[bin] >= level)
```

```
        cout << "***";
      else
        cout << "   ";
      cout << " ";
    }
    cout << endl;
  }
}
```

- *random_vector.h*

```
#include <iostream>
#include <vector>

using namespace std;

class RandomVector{
  vector<double> vect;

  public:
    RandomVector(int size, double max_val = 1);
    void print();
    double mean();
    double max();
    double min();
    void printHistogram(int bins);
};
```

- *main.cpp*

```
#include <cstdlib> // required for srand
#include <iostream>
#include "random_vector.h"

using namespace std;

int main(void) {
  srand(314159);

  RandomVector rv(20);
  rv.print();
  cout << "Mean: " << rv.mean() << endl;
  cout << "Min: " << rv.min() << endl;
  cout << "Max: " << rv.max() << endl;

  cout << "Histogram:" << endl;
  rv.printHistogram(5);
  cout << endl;

  return 0;
}
```