

PROJECT REPORT

on

Random Password Generator

(CSE III Semester Mini project)

2020-2021



Submitted to:

Mr. Umang Garg Sir
(CSE-A-III-Sem)

Guided by:

Mr. Ashook Sahoo Sir
2021

(Resource Person)

Submitted by:

Mr. Aman Gupta
Roll. No.: 19011909

CSE-A-III-Sem

Session: 2020-

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

CERTIFICATE

Certified that Mr. Aman Gupta (Roll No.- 19011909) has developed a mini project on “Random Password Generator” for the CS III Semester Mini Project Lab in Graphic Era Hill University, Dehradun. The project carried out by students is their work to the best of my knowledge.

Date: Tuesday, December 8, 2020

(Mr. Umang Garg Sir)

Project Co-ordinator

CC-CSE-B-V-Sem

(CSE Department)

GEHU Dehradun

(Mr. Ashook Sahoo Sir)

Project Guide

Resource Person

(CSE Department)

GEHU Dehradun

ACKNOWLEDGMENT

We would like to express our gratitude to The Cosmic Consciousness-Almighty Shiva Baba, the Most Beneficent and the Most Merciful, for the completion of the project.

We wish to thank our parents for their continuing support and encouragement. We also wish to thank them for providing us with the opportunity to reach this far in our studies.

We would like to thank particularly our project Guide Mr. Ashook Sahoo for his patience, support, and encouragement throughout the completion of this project and for having faith in us.

We also acknowledge teachers like Mr. Ashish Garg and Ms. Aastha Gaur who help us in developing the project.

At last but not least We are greatly indebted to all other persons who directly or indirectly helped us during this work.

Mr. Aman Gupta

Roll No.- 19011909

CSE-A-III-Sem

Session: 2020-2021

GEHU, Dehradun

TABLE OF CONTENTS

CHAPTER NO.	TITLE
-------------	-------

1.	INTRODUCTION
----	--------------

1.1	About Project
-----	---------------

1.2	C++
-----	-----

2.	PROJECT
----	---------

2.1	Module
-----	--------

2.4.1	The Naïve approach
-------	--------------------

2.4.2	Stronger methods
-------	------------------

2.4.3.	Type and Strength
--------	-------------------

3.	SNAPSHOT OF PROJECT
----	---------------------

4.	CONCLUSION
----	------------

4.1	SUMMARY
-----	---------

APPENDIX: CODE

REFERENCE

CHAPTER 1

INTRODUCTION

1.1 About Project

Random Password Generator

Write a program to generate a random password which has the following features:-

- 1) The Password should have a minimum length of 12 characters and a maximum length of 32 characters.
- 2) Password will always start with a lower case alphabet and ends with upper case alphabets.
- 3) It should have at least 2 lowercase, 2 uppercase alphabet, 1 number, 1 special character. No space is allowed in the generated password.
- 4) You should not use any dictionary of password generation.
- 5) Do not use the inbuilt function for randomization. Make your pseudo-random number generator.
- 6) GUI – there should be one button for generating a password which will be shown on the same screen inside some label, there should be one button that will copy the password on the clipboard.

RANDOM PASSWORD GENERATOR

A random password generator is software program or hardware device that takes input from a random or pseudo-random number generator and automatically generates a password. Random passwords can be generated manually, using simple sources of randomness such as dice or coins, or they can be generated using a computer.

While there are many examples of "random" password generator programs available on the Internet, generating randomness can be tricky and many programs do not generate random characters in a way that ensures strong security. A common recommendation is to use open source security tools where possible since they allow independent checks on the quality of the methods used. Note that simply generating a password at random does not ensure the password is a strong password, because it is possible, although highly unlikely, to generate an easily guessed or cracked password. In fact, there is no need at all for a password to have been produced by a perfectly random process: it just needs to be sufficiently difficult to guess.

A password generator can be part of a password manager. When a password policy enforces complex rules, it can be easier to use a password generator based on that set of rules than to manually create passwords.

C++ (why ?)

1. C++ Popularity and High Salary

C++ is one of the most popular languages in the world. It is used by some 4.4 million developers worldwide. Also, C++ Developers are quite sought after and they hold some of the most high-paying jobs in the industry with an average base pay of \$103, 035 per year.

2. C++ has Abundant Library Support

C++ has the Standard Template Library (STL) that is very useful as it helps in writing code compactly and quickly as required. It contains mainly four components i.e. algorithms, containers, functions, and iterators.

The algorithms are of different types such as sorting, searching, etc. The containers store classes to implement different data structures that are commonly used such as stacks, queues, hash tables, vectors, sets, lists, maps, etc. The functors allow the working of the associated function to be customized with the help of the parameters passed. Also, the iterators are used for working upon a sequence of values.

3. C++ has a Large Community

There is a large online community of C++ users and experts that is particularly helpful in case any support is required. There is a lot of resources like GeeksforGeeks etc. available on the internet regarding C++. Some of the other online resources for C++ include StackOverflow, cppreference.com, Standard C++, etc.

4. C++ In Databases

There are many modern day databases such as MySQL, MongoDB, MemSQL, etc. that are written in C++. This is because C++ is quite modern and it supports features like exceptions, lambda expressions, etc. Many of the databases that are written in C++ are used in almost all of the in-use applications such as YouTube, WordPress, Twitter, Facebook, etc.

5. C++ In Operating Systems

All the major operating systems such as Windows, Linux, Android, Ubuntu, iOS, etc. are written in a combination of C and C++. The Windows applications are written in C++, while Android applications are written in Java along with C/C++ with non-default run-times for C++ support. Also, C++ can be used to develop the core of the applications in iOS. In general, C or C++ are used in operating systems because of the speed and strongly typed nature of these languages.

6. C++ In Compilers

C++ is closer to the hardware level and is a comparatively low-level language. Because of this reason, it is used in many compilers as a backend programming language. An example of this is the GNU Compiler Collection (GCC) which is currently written mostly in C++ along with C.

7. C++ In Web Browsers

A lot of web browsers are developed using C++ such as Chrome, Firefox, Safari, etc. Chrome contains C++ in the rendering engine, JavaScript engine, and the UI. Firefox uses mainly in the rendering engine and a little in the UI. Safari also uses C++ in the rendering engine and JavaScript engine.

All these web browsers and more use C++, particularly in the rendering engines because it provides the required speed that is necessary for the rendering engines since they need to display the content at an accelerated rate.

The naive approach

Here are two code samples that a programmer who is not familiar with the limitations of the random number generators in standard programming libraries might implement:

//Ist Approach

```
# include <time.h>
# include <stdio.h>
# include <stdlib.h>

int
main(void)
{
    /* Length of the password */
    unsigned short int length = 8;

    /* Seed number for rand() */
    srand((unsigned int) time(0));

    /* ASCII characters 33 to 126 */
    while (length-- > 0) {
        putchar(rand() % 94 + 33);
    }

    printf("\n");

    return EXIT_SUCCESS;
}
```

//IInd Approach

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    int i = 0;
    int n = 0;
    int randomizer = 0;
    srand((time(NULL)));
    char numbers [] = "1234567890";
    char letter [] = "abcdefghijklmnopqrstuvwxyz";
    char letterr [] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    char symbols [] = "!@#$%^&*(){}[]:<?>.,/";
    printf("\nHow long password:");
    scanf("%d", &n);
    char password[n];
    randomizer = rand() % 4;
```



```

for (i=0;i<n;i++)
{
    if(randomizer == 1)
    {
        password[i] = numbers[rand() % 10];
        randomizer = rand() % 4;
        printf("%c", password[i]);
    }
    else if (randomizer == 2)
    {
        password[i] = symbols[rand() % 26];
        randomizer = rand() % 4;
        printf("%c", password[i]);
    }
    else if (randomizer == 3)
    {
        password[i] = lettterr[rand() % 26];
        randomizer = rand() % 4;
        printf("%c", password[i]);
    }
    else
    {
        password[i] = letter[rand() % 21];
        randomizer = rand() % 4;
        printf("%c", password[i]);
    }
}
return main();
}

```

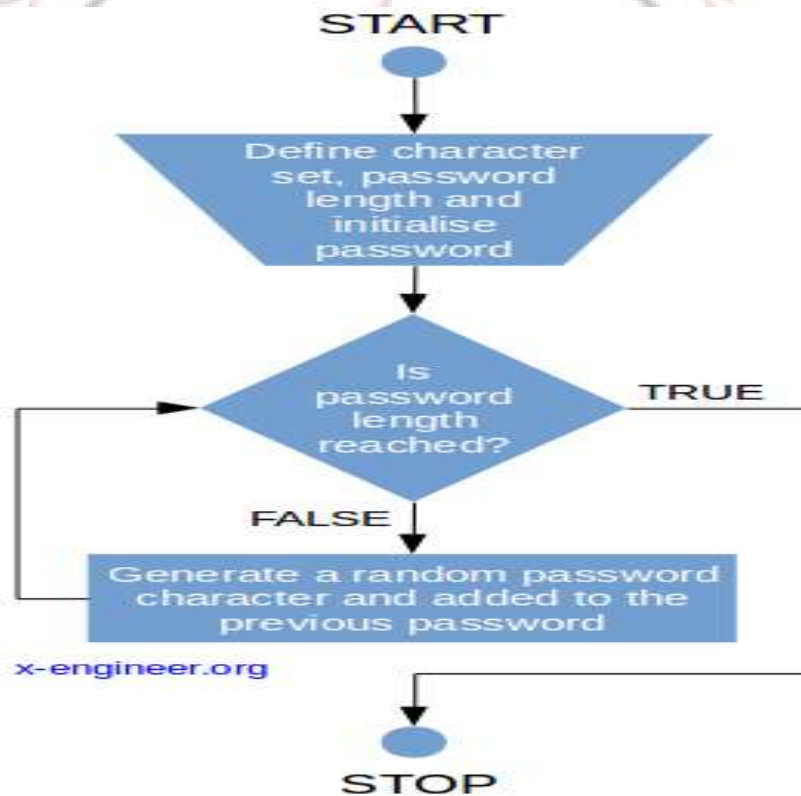
In this case, the standard C function `rand`, which is a pseudo-random number generator, is initially seeded using the C functions `time`, but later iterations use `rand` instead. According to the ANSI C standard, `time` returns a value of type `time_t`, which is implementation-defined, but most commonly a 32-bit integer containing the current number of seconds since January 1, 1970 (see: Unix time). There are about 31 million seconds in a year, so an attacker who knows the year (a simple matter in situations where frequent password changes are mandated by password policy) and the process ID that the password was generated with, faces a relatively small number, by cryptographic standards, of choices to test. If the attacker knows more accurately when the password was generated, he faces an even smaller number of candidates to test – a serious flaw in this implementation.

In situations where the attacker can obtain an encrypted version of the password, such testing can be performed rapidly enough so that a few million trial passwords can be checked in a matter of seconds.

Type and strength of password generated

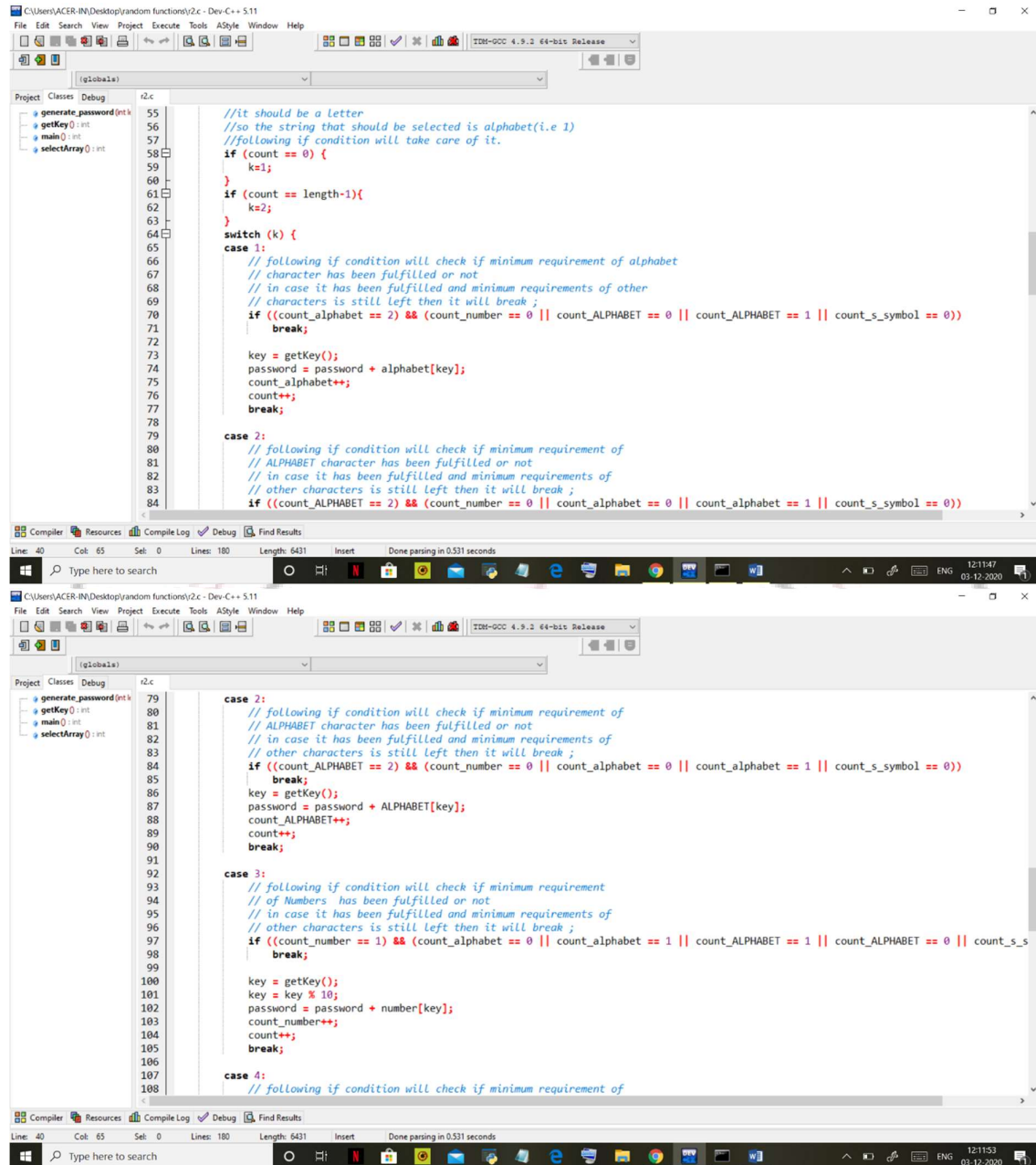
Random password generators normally output a string of symbols of specified length. These can be individual characters from some character set, syllables designed to form pronounceable passwords, or words from some word list to form a passphrase. The program can be customized to ensure the resulting password complies with the local password policy, say by always producing a mix of letters, numbers and special characters. Such policies typically reduce strength slightly below the formula that follows, because symbols are no longer independently produced.

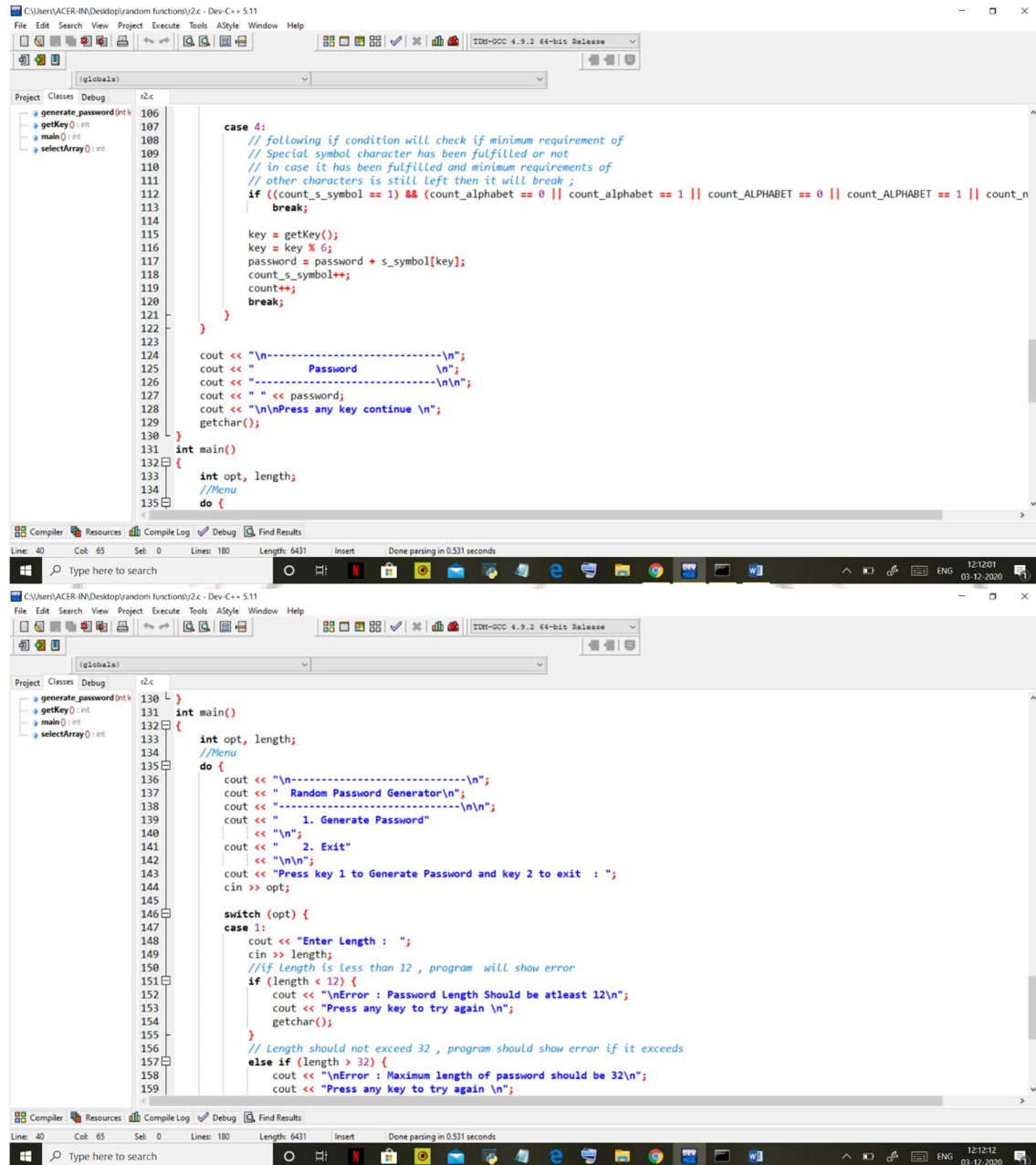
BASIC FLOWCHART FOR RANDOM PASSWORD GENERATOR



SNAPSHOT OF PROJECT

```
1 #include<iostream>
2 #include<stdlib.h>
3 #include<time.h>
4 using namespace std;
5
6 //selectArray is a utility function that is used to
7 //randomly generate a integer in the range 1 to 4 (both inclusive)
8 int selectArray()
9 {
10     srand(time(NULL));
11     int i = rand() % 5;
12     if (i == 0)
13         i++;
14     return i;
15 }
16
17 //getKey() is another utility function that is used to randomly generate
18 //an integer in the range 0 to 25 (both inclusive)
19 int getKey()
20 {
21     srand(time(NULL));
22     int key = rand() % 26;
23     return key;
24 }
25
26 void generate_password(int length)
27 {
28     //Intializing result string password as NULL.
29     string password = "";
30
31     //Strings whose characters will be used to build password
32     string alphabet = "abcdefghijklmnopqrstuvwxyz";
33     string ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
34     string s_symbol = "!@#$%&*";
35     string number = "0123456789";
36
37     //initializing local variables
38     int key, count_alphabet = 0, count_ALPHABET = 0, count_number = 0, count_s_symbol = 0;
39
40     //Count will store the Length of the password being created,
41     //initially this will be zero(0)
42     int count = 0;
43     while (count < length) {
44         // selectArray() function will return a number 1 to 4
45         // and will use to select one of the above defined string
46         //(i.e alphabet or ALPHABET or s_symbol or number )
47         // 1 is for string alphabet
48         // 2 is for string ALPHABET
49         // 3 is for string number
50         // and 4 is for string s_symbol
51
52         int k = selectArray();
53
54         //for the first character of password it is mentioned that,
55         //it should be a letter
56         //so the string that should be selected is alphabet(i.e 1)
57         //following if condition will take care of it.
```





```
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159

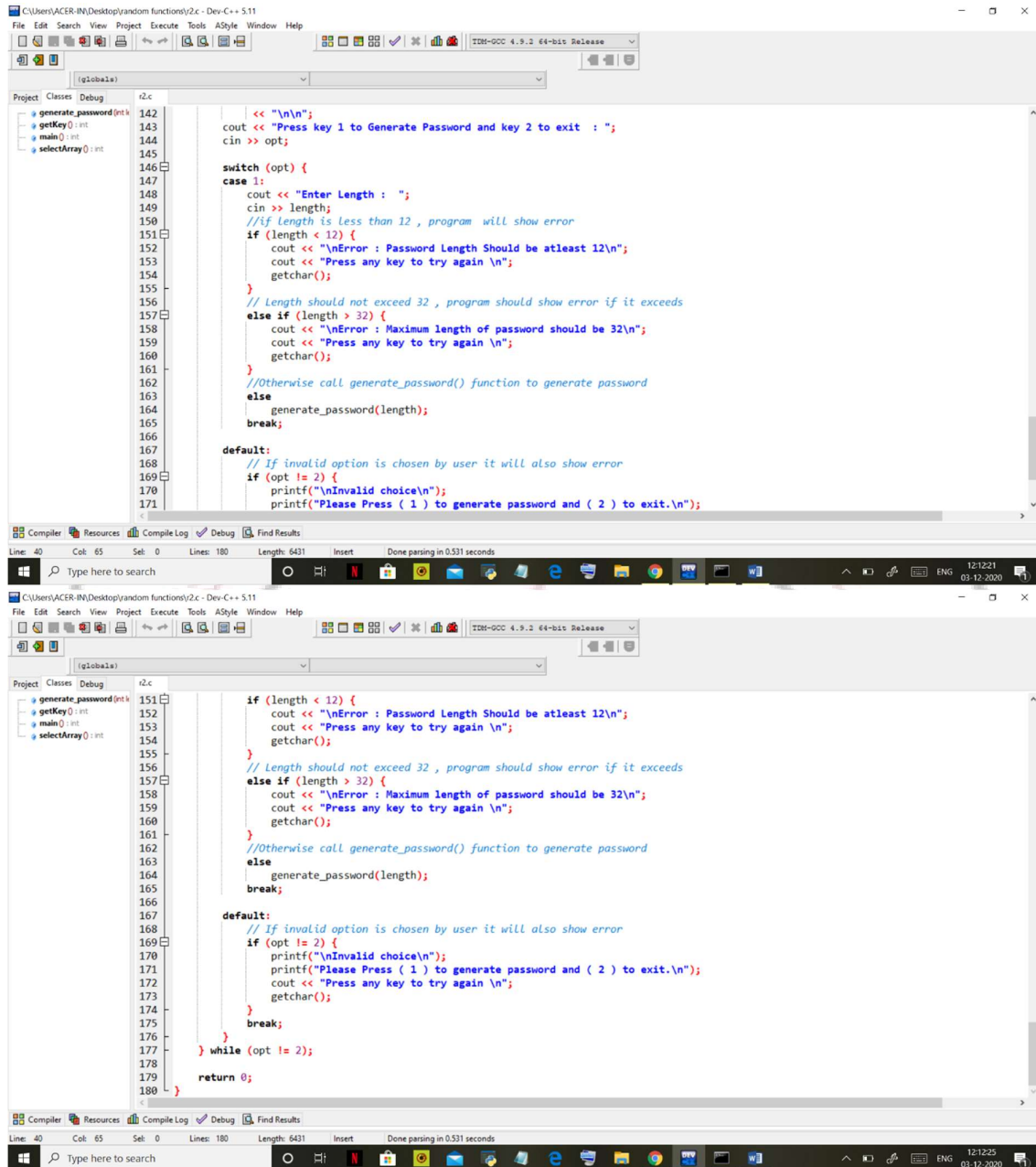
case 4:
    // following if condition will check if minimum requirement of
    // Special symbol character has been fulfilled or not
    // in case it has been fulfilled and minimum requirements of
    // other characters is still left then it will break ;
    if ((count_s_symbol == 1) && (count_alphabet == 0 || count_alphabet == 1 || count_ALPHABET == 0 || count_ALPHABET == 1 || count_n
        break;

    key = getKey();
    key = key % 6;
    password = password + s_symbol[key];
    count_s_symbol++;
    count++;
    break;
}

cout << "\n-----\n";
cout << "      Password      \n";
cout << "-----\n\n";
cout << "<< password;
cout << "\n\nPress any key continue \n";
getchar();

int main()
{
    int opt, length;
    //Menu
    do {
        cout << "\n-----\n";
        cout << "      Random Password Generator\n";
        cout << "-----\n\n";
        cout << "      1. Generate Password"
        cout << "\n";
        cout << "      2. Exit"
        cout << "\n\n";
        cout << "Press key 1 to Generate Password and key 2 to exit : ";
        cin >> opt;

        switch (opt) {
        case 1:
            cout << "Enter Length : ";
            cin >> length;
            //if length is Less than 12 , program will show error
            if (length < 12) {
                cout << "\nError : Password Length Should be atleast 12\n";
                cout << "Press any key to try again \n";
                getchar();
            }
            // Length should not exceed 32 , program should show error if it exceeds
            else if (length > 32) {
                cout << "\nError : Maximum length of password should be 32\n";
                cout << "Press any key to try again \n";
            }
        }
    } while (opt != 2);
}
```



Snapshot of output

```
Command Prompt
-----
Random Password Generator
-----
1. Generate Password
2. Exit

Press key 1 to Generate Password and key 2 to exit : 1
Enter Length : 10
Error : Password Length Should be atleast 12
Press any key to try again

-----
Random Password Generator
-----
1. Generate Password
2. Exit

Press key 1 to Generate Password and key 2 to exit : 13
Invalid choice
Please Press ( 1 ) to generate password and ( 2 ) to exit.
Press any key to try again

-----
Random Password Generator
-----
1. Generate Password
2. Exit

Press key 1 to Generate Password and key 2 to exit : 1
Enter Length : 13

-----
Password
-----

Type here to search

Command Prompt
-----
2. Exit

Press key 1 to Generate Password and key 2 to exit : 13
Invalid choice
Please Press ( 1 ) to generate password and ( 2 ) to exit.
Press any key to try again

-----
Random Password Generator
-----
1. Generate Password
2. Exit

Press key 1 to Generate Password and key 2 to exit : 1
Enter Length : 13

-----
Password
-----

oo8UU444444E

Press any key continue

-----
Random Password Generator
-----
1. Generate Password
2. Exit

Press key 1 to Generate Password and key 2 to exit : 2
C:\Users\ACER-IN\Desktop\random functions>
```


CONCLUSION

In today's modern technological age, everybody has various devices such as desktop computers, laptops, tablets, iPads and smart phone. Because nobody wants their personal information stolen, this means people need passwords to protect everything they use.

This doesn't merely mean protecting the device itself from being hacked. It also means preventing hackers getting your information from websites.

Social medial sites such as Facebook, LinkedIn, Twitter and many others are used daily by millions of people all around the world. Some people use the same password for them all. Online banking, running websites, using emails and many other things all need protection from prying eyes and so you need passwords for them all.

Using **strong passwords** can help shield against traditional **password** attacks such as dictionary, rainbow tables, or brute-force attacks. **You need strong passwords** so these **strong password generator** tools **will** help keep **you** safe from being attacked online.



Problem Statement:

Write a menu driven program to generate password randomly

constraint:

1. password should consist of
 - lowercase Alphabet - a to z
 - UpperCase Alphabet - A to Z
 - Number - 0 to 9
 - Special Symbol - !, @, #, \$, %, &
2. Password length should be
 - Minimum - 12
 - Maximum - 32
3. Password should begin with a letter (lowercase)
4. Password should contain at least 2 lowercase letter , 2 uppercase letter, 1 number , and 1 special symbol
5. Don't make use of any library function like rand() or srand().
6. Each time generated password should be unique.



APPENDIX

Code

Program to generate random password

```
#include<iostream>
#include<stdlib.h>
#include<time.h>
using namespace std;

//selectArray is a utility function that is used to
//randomly generate a integer in the range 1 to 4 (both inclusive)
int selectArray()
{
    srand(time(NULL));
    int i = rand() % 5;
    if (i == 0)
        i++;
    return i;
}

//getKey() is another utility function that is used to randomly generate
//an integer in the range 0 to 25 (both inclusive)
int getKey()
{
    srand(time(NULL));
```

```
int key = rand() % 26;

return key;
}

void generate_password(int length)
{
    //Intializing result string password as NULL.
    string password = "";

    //Strings whose characters will be used to build password
    string alphabet = "abcdefghijklmnopqrstuvwxyz";
    string ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    string s_symbol = "!@#$%&";
    string number = "0123456789";

    //initializing local variables
    int key, count_alphabet = 0, count_ALPHABET = 0, count_number = 0, count_s_symbol = 0;

    //Count will store the length of the password being created,
    //initially this will be zero(0)
    int count = 0;
    while (count < length) {
        // selectArray() function will return a number 1 to 4
        // and will use to select one of the above defined string
        //(i.e alphabet or ALPHABET or s_symbol or number )
        // 1 is for string alphabet
        // 2 is for string ALPHABET
```

```
// 3 is for string number

// and 4 is for string s_symbol

int k = selectArray();

//for the first character of password it is mentioned that,
//it should be a letter
//so the string that should be selected is alphabet (i.e 1)
//following if condition will take care of it.
if (count == 0) {
    k=1;
}
if (count == length-1){
    k=2;
}
switch (k) {
case 1:
    // following if condition will check if minimum requirement of alphabet
    // character has been fulfilled or not
    // in case it has been fulfilled and minimum requirements of other
    // characters is still left then it will break ;
    if ((count_alphabet == 2) && (count_number == 0 || count_ALPHABET == 0 ||
count_ALPHABET == 1 || count_s_symbol == 0))
        break;

    key = getKey();
    password = password + alphabet[key];
```

```
count_alphabet++;
```

```
count++;
```

```
break;
```

case 2:

```
// following if condition will check if minimum requirement of
```

```
// ALPHABET character has been fulfilled or not
```

```
// in case it has been fulfilled and minimum requirements of
```

```
// other characters is still left then it will break ;
```

```
if ((count_ALPHABET == 2) && (count_number == 0 || count_alphabet == 0 ||  
count_alphabet == 1 || count_s_symbol == 0))
```

```
break;
```

```
key = getKey();
```

```
password = password + ALPHABET[key];
```

```
count_ALPHABET++;
```

```
count++;
```

```
break;
```

case 3:

```
// following if condition will check if minimum requirement
```

```
// of Numbers has been fulfilled or not
```

```
// in case it has been fulfilled and minimum requirements of
```

```
// other characters is still left then it will break ;
```

```
if ((count_number == 1) && (count_alphabet == 0 || count_alphabet == 1 ||  
count_ALPHABET == 1 || count_ALPHABET == 0 || count_s_symbol == 0))
```

```
break;
```

```
key = getKey();
```

```

key = key % 10;

password = password + number[key];

count_number++;

count++;

break;

case 4:

    // following if condition will check if minimum requirement of
    // Special symbol character has been fulfilled or not
    // in case it has been fulfilled and minimum requirements of
    // other characters is still left then it will break ;
    if ((count_s_symbol == 1) && (count_alphabet == 0 || count_alphabet == 1 ||
count_ALPHABET == 0 || count_ALPHABET == 1 || count_number == 0))
        break;

    key = getKey();
    key = key % 6;
    password = password + s_symbol[key];
    count_s_symbol++;
    count++;
    break;
}
}

cout << "\n-----\n";

cout << "    Password    \n";

cout << "-----\n\n";

```

```
cout << " " << password;

cout << "\n\nPress any key continue \n";

getchar();
}

int main()
{
    int opt, length;

    //Menu
    do {
        cout << "\n-----\n";
        cout << " Random Password Generator\n";
        cout << "-----\n\n";
        cout << " 1. Generate Password"
            << "\n";
        cout << " 2. Exit"
            << "\n\n";
        cout << "Press key 1 to Generate Password and key 2 to exit : ";
        cin >> opt;

        switch (opt) {
            case 1:
                cout << "Enter Length : ";
                cin >> length;

                //if length is less than 12 , program will show error
                if (length < 12) {
                    cout << "\nError : Password Length Should be atleast 12\n";
                    cout << "Press any key to try again \n";
                }
            }
        }
    }
```

```
        getchar();
    }

    // Length should not exceed 32 , program should show error if it exceeds
    else if (length > 32) {

        cout << "\nError : Maximum length of password should be 32\n";

        cout << "Press any key to try again \n";

        getchar();
    }

    //Otherwise call generate_password() function to generate password
    else

        generate_password(length);

    break;

default:

    // If invalid option is chosen by user it will also show error
    if (opt != 2) {

        printf("\nInvalid choice\n");

        printf("Please Press ( 1 ) to generate password and ( 2 ) to exit.\n");

        cout << "Press any key to try again \n";

        getchar();

    }

    break;

}

} while (opt != 2);

return 0;

}
```


Output:

Random Password Generator

1. Generate Password
2. Exit

Press key 1 to Generate Password and key 2 to exit : 1

Enter Length : 10

Error : Password Length Should be atleast 12

Press any key to try again

Random Password Generator

1. Generate Password
2. Exit

Press key 1 to Generate Password and key 2 to exit : 13

Invalid choice

Please Press (1) to generate password and (2) to exit.

Press any key to try again

Random Password Generator

1. Generate Password
2. Exit

Press key 1 to Generate Password and key 2 to exit : 1

Enter Length : 13

Password

oo&UU4444444E

Press any key continue

Random Password Generator

1. Generate Password
2. Exit

Press key 1 to Generate Password and key 2 to exit : 2



REFERENCE

1. codespeedy.com
2. RANDOM PASSWORD GENERATOR WIKI
3. Stack overflow
4. You Tube channels
5. ANSI C
6. Google
7. Greek for Greeks

