

Project Report

Password Generator

Submitting to MICRO IT

in

**Department of Computer Science and Engineering with Artificial Intelligence and
Machine Learning**

By

G. Anil (241U1R2010)



AURORA HIGHER EDUCATION AND RESEARCH ACADEMY

(Deemed- to- be- University) Parvathapur, Uppal, Hyderabad-500 098

Introduction:

In the digital world, we use passwords to keep our accounts and information safe. If a password is easy to guess, it can lead to hacking and data loss. That's why strong passwords are very important.

This project is a Password Generator made in C programming language. It helps users create strong and random passwords. The user can choose how long the password should be and what types of characters it should include like small letters, capital letters, numbers, and special symbols.

The program then mixes the selected characters and creates a password that is hard to guess. This project also helps us learn basic programming skills like taking user input, working with strings, using random numbers, and writing conditions in C.

Code:

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

#define MAX_LENGTH 100

// Character sets
const char lowercase[] = "abcdefghijklmnopqrstuvwxyz";
const char uppercase[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
const char digits[] = "0123456789";
const char special[] = "!@#$%^&*()-_=[]{}|;:'.<>?/";

void generatePassword(int length, int useLower, int useUpper, int useDigits, int useSpecial) {
    char allChars[256] = ""; // Enough to store combined sets
    char password[MAX_LENGTH];
    int i, index = 0;

    // Add selected character sets to allChars
    if (useLower) strcat(allChars, lowercase);
    if (useUpper) strcat(allChars, uppercase);
    if (useDigits) strcat(allChars, digits);
    if (useSpecial) strcat(allChars, special);

    int allCharsLen = strlen(allChars);

    if (allCharsLen == 0) {
        printf("Error: No character types selected.\n");
        return;
    }
}
```

```

// Seed random number generator
srand(time(NULL));

// Ensure at least one character from each selected type is included
if (useLower) password[index++] = lowercase[rand() % strlen(lowercase)];
if (useUpper) password[index++] = uppercase[rand() % strlen(uppercase)];
if (useDigits) password[index++] = digits[rand() % strlen(digits)];
if (useSpecial) password[index++] = special[rand() % strlen(special)];

// Fill the rest of the password randomly
for (i = index; i < length; i++) {
    password[i] = allChars[rand() % allCharsLen];
}

password[length] = '\0'; // End the string

// Shuffle the password so first characters aren't always from selected sets
for (i = 0; i < length; i++) {
    int j = rand() % length;
    char temp = password[i];
    password[i] = password[j];
    password[j] = temp;
}

printf("\nYour generated password is: %s\n", password);
}

int main() {
    int length;
    int useLower, useUpper, useDigits, useSpecial;

```

```

printf("=== Password Generator in C ===\n\n");
printf("Enter password length (1 to %d): ", MAX_LENGTH - 1);
scanf("%d", &length);

if (length <= 0 || length >= MAX_LENGTH) {
    printf("Invalid password length.\n");
    return 1;
}

printf("Include lowercase letters? (1 = Yes, 0 = No): ");
scanf("%d", &useLower);
printf("Include uppercase letters? (1 = Yes, 0 = No): ");
scanf("%d", &useUpper);
printf("Include digits? (1 = Yes, 0 = No): ");
scanf("%d", &useDigits);
printf("Include special characters? (1 = Yes, 0 = No): ");
scanf("%d", &useSpecial);

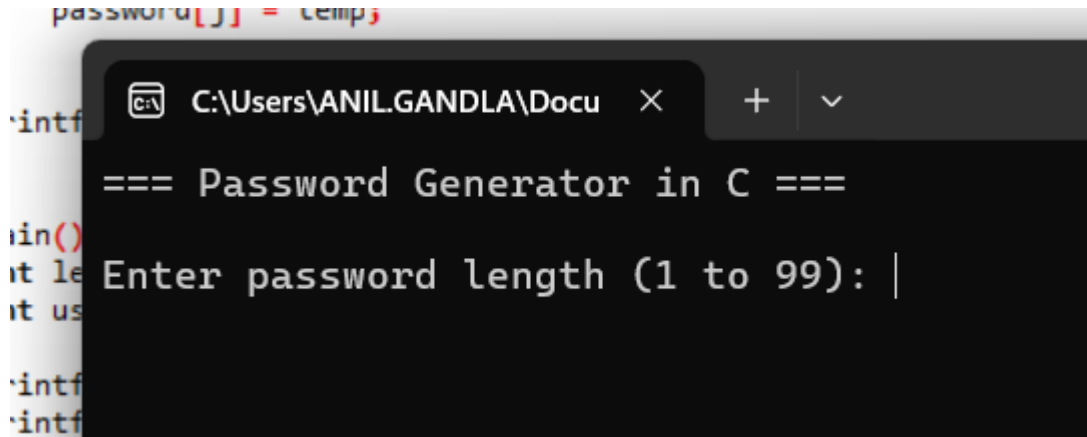
// Check if the user selected enough character types for the password length
int minRequired = useLower + useUpper + useDigits + useSpecial;
if (minRequired > length) {
    printf("Error: Password length is too short for selected character types.\n");
    return 1;
}

generatePassword(length, useLower, useUpper, useDigits, useSpecial);

return 0;
}

```

Output:



```
password[] = temp;

intf

C:\Users\ANIL.GANDLA\Docu  X  +  v

=== Password Generator in C ===

Enter password length (1 to 99): |

intf
intf
```

Code Explanation:

◆ Step 1: Include Header Files

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

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

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

```
#include <string.h>
```

These are required to:

- Print messages (stdio.h)
- Use random numbers and memory functions (stdlib.h)
- Get the current time for randomness (time.h)
- Work with strings (string.h)

◆ Step 2: Define Maximum Password Length

```
#define MAX_LENGTH 100
```

This says the longest password we allow is **99 characters** (+1 for ending \0).

◆ Step 3: Define Character Sets

```
const char lowercase[] = "abcdefghijklmnopqrstuvwxyz";
```

```
const char uppercase[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
const char digits[] = "0123456789";
```

```
const char special[] = "!@#$%^&*()-_+=[]{}|;:'.<>?/";
```

These are the **different types of characters** we can use in the password.

◆ Step 4: The generatePassword() Function

This function actually **creates the password**.

```
void generatePassword(int length, int useLower, int useUpper, int useDigits, int useSpecial)
```

It takes these inputs:

- length: how long the password should be
- useLower, useUpper, useDigits, useSpecial: 1 if user wants that type, 0 if not

Inside the Function:

```
char allChars[256] = ""; // Store all selected characters
```

```
char password[MAX_LENGTH]; // Store the final password
```

```
int i, index = 0;
```

◆ Step 5: Add Selected Characters

```
if (useLower) strcat(allChars, lowercase);
```

```
if (useUpper) strcat(allChars, uppercase);
```

```
if (useDigits) strcat(allChars, digits);
```

```
if (useSpecial) strcat(allChars, special);
```

This adds selected character types to allChars.

If the user selects lowercase and digits, allChars will look like:

```
"abcdefghijklmnopqrstuvwxyz0123456789"
```

◆ Step 6: Check for Error (No Types Selected)

```
if (strlen(allChars) == 0) {
```

```
    printf("Error: No character types selected.\n");
```

```
    return;
```

```
}
```

If user didn't choose any type, show an error and stop.

◆ Step 7: Seed Random Number Generator

```
srand(time(NULL));
```

This makes the random numbers different each time the program runs.

◇ Step 8: Make Sure At Least One from Each Selected Type is Included

```
if (useLower) password[index++] = lowercase[rand() % strlen(lowercase)];
```

```
if (useUpper) password[index++] = uppercase[rand() % strlen(uppercase)];
```

```
if (useDigits) password[index++] = digits[rand() % strlen(digits)];
```

```
if (useSpecial) password[index++] = special[rand() % strlen(special)];
```

We add **one character** from each selected type to make sure they are included.

◇ Step 9: Fill the Rest of the Password

```
for (i = index; i < length; i++) {  
    password[i] = allChars[rand() % strlen(allChars)];  
}
```

This adds **random characters** from the selected group to complete the password.

◇ Step 10: End the String

```
password[length] = '\0';
```

Adds the **null character** to end the string.

◇ Step 11: Shuffle the Password

```
for (i = 0; i < length; i++) {  
    int j = rand() % length;  
    char temp = password[i];  
    password[i] = password[j];  
    password[j] = temp;  
}
```

Mixes the characters randomly so the first few characters are not always fixed.

◇ Step 12: Show the Password

```
printf("\nYour generated password is: %s\n", password);
```


◇ Step 13: Main Function

```
int main()
```

This is where the program starts.

◇ Step 14: Ask the User for Input

```
printf("Enter password length: ");
```

```
scanf("%d", &length);
```

```
// Ask for each character type
```

The user is asked:

- How long should the password be?
- Include lowercase? (1 or 0)
- Include uppercase? (1 or 0)
- Include numbers? (1 or 0)
- Include symbols? (1 or 0)

◇ Step 15: Check Length and Character Type Count

```
int minRequired = useLower + useUpper + useDigits + useSpecial;
```

```
if (minRequired > length) {
```

```
    printf("Error: Password length is too short.\n");
```

```
    return 1;
```

```
}
```

If user chooses 4 types, the length must be at least 4.

◇ Step 16: Generate the Password

```
generatePassword(length, useLower, useUpper, useDigits, useSpecial);
```

This calls the function to create and print the password.

