C Program to Check Password Validity

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

// Function to check if the password meets all criteria
int isValidPassword(char password[]) {
    int length = strlen(password);

    // Check if the length is between 8 and 20 characters
    if (length < 8 || length > 20) {
        printf("Error: Password must be between 8 and 20 characters.\n");
        return 0;
    }

    int hasUpper = 0, hasLower = 0, hasDigit = 0, hasSpecial = 0;

    // Iterate through each character in the password
    for (int i = 0; i < length; i++) {
        if (isupper(password[i])) {
            hasUpper = 1; // Set flag if uppercase letter is found
        } else if (islower(password[i])) {
            hasLower = 1; // Set flag if lowercase letter is found
        } else if (isdigit(password[i])) {
            hasDigit = 1; // Set flag if digit is found
        } else if (ispunct(password[i])) {
            hasSpecial = 1; // Set flag if special character is found
        } else if (password[i] == ' ') {
            printf("Error: Password should not contain spaces.\n");
            return 0;
        }
```

```
    }

    // Check if the password meets all the required conditions
    if (!hasUpper) {
        printf("Error: Password must contain at least one uppercase letter.\n");
        return 0;
    }
    if (!hasLower) {
        printf("Error: Password must contain at least one lowercase letter.\n");
        return 0;
    }
    if (!hasDigit) {
        printf("Error: Password must contain at least one digit.\n");
        return 0;
    }
    if (!hasSpecial) {
        printf("Error: Password must contain at least one special character.\n");
        return 0;
    }

    // If all checks pass, the password is valid
    return 1;
}

int main() {
    char password[100];

    // Input the password
    printf("Enter the password: ");
    fgets(password, sizeof(password), stdin);

    // Remove the trailing newline character if present
    password[strcspn(password, "\n")] = '\0';
```

```
    // Validate the password
    if (isValidPassword(password)) {
        printf("\nPassword is valid!\n");
    } else {
        printf("\nPassword is invalid.\n");
    }

    return 0;
}
```

Explanation of the Program:

1. **Password Validation Function** (`isValidPassword`):
    - This function checks if the password meets the specified criteria:
        - **Length**: The password length must be between 8 and 20 characters.
        - **Uppercase Letter**: The function checks if at least one uppercase letter exists.
        - **Lowercase Letter**: The function checks if at least one lowercase letter exists.
        - **Digit**: The function checks if at least one digit exists.
        - **Special Character**: The function checks if at least one special character exists (using `ispunct` function).
        - **No Spaces**: The password should not contain any spaces (this check is done using a simple conditional check for `' '`).
2. **Main Function**:
    - Prompts the user to enter a password.
    - The `fgets()` function is used to safely read the input, and `strcspn()` is used to remove any trailing newline character from the input string.

- The program calls `isValidPassword` to check if the entered password meets the required criteria.
- Based on the validation result, it prints whether the password is valid or invalid.

Logic of the Program:

1. **Input**:
   - The program asks for the user to input a password. The user enters the password as a string.
2. **Password Length Check**:
   - The function checks if the length of the password is within the range of 8 to 20 characters. If not, an error is printed, and the password is considered invalid.
3. **Character Analysis**:
   - The program iterates through each character of the password and checks for:
     - **Uppercase Letter**: Using the `isupper()` function.
     - **Lowercase Letter**: Using the `islower()` function.
     - **Digit**: Using the `isdigit()` function.
     - **Special Character**: Using the `ispunct()` function, which checks for punctuation and special symbols.
     - **Spaces**: The program checks for spaces and reports an error if any are found.
4. **Error Handling**:
   - If any of the criteria fail, the program prints an error message and returns `0` (indicating invalid password).
   - If all criteria are met, the password is considered valid, and the function returns `1`.
5. **Output**:
   - If the password is valid, a success message is printed: `"Password is valid!"`.
   - If any criteria fail, an error message is printed explaining the reason why the password is invalid.

Sample Output:

Valid Password:

Enter the password: Passw0rd!
Password is valid!

Invalid Password (missing uppercase letter):

Enter the password: password1!
Error: Password must contain at least one uppercase letter.

Password is invalid.

Invalid Password (contains spaces):

Enter the password: Pass word1!
Error: Password should not contain spaces.

Password is invalid.

Invalid Password (missing special character):

Enter the password: Password123
Error: Password must contain at least one special character.

Password is invalid.

Summary of the Logic:

1. **Input the password** from the user.
2. **Check the password length** to ensure it is between 8 and 20 characters.
3. **Iterate through each character** of the password and:
    - Check if it contains at least one uppercase letter.
    - Check if it contains at least one lowercase letter.

    - Check if it contains at least one digit.
    - Check if it contains at least one special character.
4. **Check for spaces** to ensure there are no spaces in the password.
5. **Print appropriate messages** based on whether the password meets the criteria or not.

Extension Ideas:

- You could extend this program to check for more complex password rules, such as:
    - Checking if the password contains both letters and numbers.
    - Preventing the use of commonly used passwords.
    - Adding more specific special characters.