

## main.cxx

```
1  #include <iostream>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5  #include <string>
6  #include <cstring>
7  #include <passgen.hxx>
8  using namespace PassGen;
9
10
11 void printhelp() {
12     printf("APCSPCreateTask - Random Password Generator\n\n");
13     printf("[Usage]: APCSPCreateTask [-A -a -n -s] -l <length>\n\n");
14     printf("[Arguments]:\n\n");
15     printf("\t-A : include upper case alphabets in password\n\n");
16     printf("\t-a : include lower case alphabets in password\n\n");
17     printf("\t-n : include numbers in password\n\n");
18     printf("\t-s : include special characters in password\n\n");
19     printf("\t-l <number> : set the length of the password\n\n");
20     printf("\t-h : print this help\n\n");
21 }
22
23 bool checkFlags(bool up, bool low, bool num, bool spec) {
24     int count = 0;
25     if (up == true) {count++;}
26     if (low == true) {count++;}
27     if (num == true) {count++;}
28     if (spec == true) {count++;}
29     return count > 0 ? true : false;
30 }
31
32 int main(int argc, char** argv) {
33     // flags for program arguments
34     bool upAlphaFlag = false;
35     bool lowAlphaFlag = false;
36     bool numFlag = false;
37     bool specialCharFlag = false;
38     int length = -1;
39     int arg;
40
41     std::string input;
42
43     while ((arg = getopt (argc, argv, "Aanshl:")) != -1) {
44         switch (arg) {
45             case 'A':
46                 upAlphaFlag = true;
47                 break;
48             case 'a':
49                 lowAlphaFlag = true;
50                 break;
51             case 'n':
52                 numFlag = true;
53                 break;
54             case 's':
```

```

55         specialCharFlag = true;
56         break;
57     case 'l':
58         length = std::atoi(optarg);
59         break;
60     case 'h':
61         printhelp();
62         return 0;
63     case '?':
64         if (optopt == 'l') {
65             printf("Error: No length specified\n");
66         } else if (isprint(optopt)){
67             printf("Error: Unknown option: -%c\n", optopt);
68         } else {
69             printf("Error: Unknown value: -%c\n", optopt);
70         }
71         return 1;
72     default:
73         printhelp();
74         abort();
75 }
76 }
77
78 if (length < 0 && checkFlags(upAlphaFlag, lowAlphaFlag, numFlag,
79     specialCharFlag) == false) {
80     printhelp();
81     printf("\nError: No option specified\n");
82     return 1;
83 }
84
85 if (length < 0) {
86     printhelp();
87     printf("\nError: Length not specified\n");
88     return 1;
89 }
90
91 if (checkFlags(upAlphaFlag, lowAlphaFlag, numFlag,
92     specialCharFlag) == false) {
93     printhelp();
94     printf("\nError: No character flag(s) specified\n");
95     return 1;
96 }
97
98 if (upAlphaFlag == true) {input += getUpperAlpha();}
99 if (lowAlphaFlag == true) {input += getLowerAlpha();}
100 if (numFlag == true) {input += getNumber();}
101 if (specialCharFlag == true) {input += getSpecialChars();}
102
103 char *cInput = new char[input.length() + 1];
104 strcpy(cInput, input.c_str());
105 char *out = passGen(cInput, length);
106 std::cout << out << std::endl;
107
108 return 0;
109 }

```

passgen.hxx

```

1 #ifndef PASSGEN
2 #define PASSGEN
3
4 #include <stdlib.h>
5 #include <stdio.h>
6
7 namespace PassGen {
8     char* getLowerAlpha();
9     char* getUpperAlpha();
10    char* getNumber();
11    char* getSpecialChars();
12    char* passGen(char* charList, int len);
13 }
14
15 #endif // PASSGEN

```

### passgen.cxx

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 #include <utils.hxx>
6 #include <passgen.hxx>
7
8 // getSpecialChars - get the lower case alphabets
9 // void : takes nothing
10 // return (char*) : the string with all lower case alphabets in
    standard ASCII
11 char* PassGen::getLowerAlpha() {
12     char* output = new char[26]; // 26 letters
13     if (output == NULL) {return 0;} // check if memory allocation
        is failed
14     int offset = 97; // 97th letter in ASCII (a)
15     for (int i = 0; i < 26; i++) {
16         output[i] = offset + i;
17     }
18     return output;
19 }
20
21 // getUpperAlpha - get the upper case alphabets
22 // void : takes nothing
23 // return (char*) : the string with all upper case alphabets in
    standard ASCII
24 char* PassGen::getUpperAlpha() {
25     char* output = new char[26]; // 26 letters
26     if (output == NULL) {return 0;} // check if memory allocation
        is failed
27     int offset = 65; // 65th letter in ASCII (A)
28     for (int i = 0; i < 26; i++) {
29         output[i] = offset + i;
30     }
31     return output;
32 }
33

```

```

34 // getSpecialChars - get the numbers
35 // void : takes nothing
36 // return (char*) : the string with all numbers in standard ASCII
37 char* PassGen::getNumber() {
38     char* output = new char[10]; // 10 letters
39     if (output == NULL) {return 0;} // check if memory allocation
40     // is failed
41     int offset = 48; // 48th letter in ASCII (0)
42     for (int i = 0; i < 10; i++) {
43         output[i] = offset + i;
44     }
45     return output;
46 }
47
48 // getSpecialChars - get the special characters
49 // void : takes nothing
50 // return (char*) : the string with all special characters in
51 // standard ASCII
52 char* PassGen::getSpecialChars() {
53     char* output = new char[42]; // 42 symbols
54     if (output == NULL) {return 0;} // check if memory allocation
55     // is failed
56     int offset = 33; // 33rd letter in ASCII (!)
57     int listOffset = 0;
58     int i;
59     // ASCII range of 33 - 64 (32 symbols)
60     for (i = 0; i < 32; i++) {
61         output[i] = offset + i;
62     }
63     listOffset = 32;
64     offset = 91;
65     // ASCII range of 91 - 96 (6 symbols)
66     for (i = 0; i < 6; i++) {
67         output[i + listOffset] = offset + i;
68     }
69     listOffset = 38;
70     offset = 123;
71     // ASCII range of 123 - 126 (4 symbols)
72     for (i = 0; i < 4; i++) {
73         output[i + listOffset] = offset + i;
74     }
75     return output;
76 }
77
78 // passGen - Password Generator
79 // charList (char*) : list of char to be used in password
80 // generation
81 // len (int) : length of password
82 // return (char*) : the generated password
83 char* PassGen::passGen(char *charList, const int len) {
84     std::srand(time(nullptr));
85     unsigned int index;
86     char* output = new char[len+1]; // length of password + 1
87     // terminating char
88     if (output == NULL) {return 0;} // return 0 on the failiure of
89     // memory allocation

```

```

85     for (int i = 0; i <= len; i++) {
86         if (i == len) {output[i] = charList[strSize(charList)];}
87         else {
88             index = std::rand()%(strSize(charList));
89             output[i] = charList[index];
90         }
91     }
92     return output;
93 }

```

utils.hxx

```

1  #ifndef UTILS
2  #define UTILS
3
4  int strSize(char* a) {
5      int out = 0;
6      int i = 0;
7      while (a[i] != 0) {
8          i++;
9          out++;
10     }
11     return out;
12 }
13
14 #endif // UTILS

```