Karakterláncok (String)

1. Karaktertömb hosszának meghatározása A feladat: Írj egy programot, amely egy karaktertömb hosszát határozza meg, és kiírja azt!

```
#include <stdio.h>
#include <string.h>
int main() {
    char szoveg[] = "Hello, world!";
    int hossz = strlen(szoveg);
    printf("A karaktertömb hossza: %d\n", hossz);
}
```

2. Karakterek számolása egy karaktertömbben A feladat: Írj egy programot, amely megszámolja, hány 'e' karakter található egy adott karaktertömbben!

```
#include <stdio.h>
int main() {
    char szoveg[] = "Hello, world!";
    int count = 0;
    for (int i = 0; szoveg[i] != '\0'; i++) {
        if (szoveg[i] == 'e') {
            count++;
        }
    }
    printf("Az 'e' karakterek száma: %d\n", count);
}
```

3. Két karaktertömb összefűzése A feladat: Írj egy programot, amely két karaktertömböt fűz össze, majd kiírja az eredményt!

```
#include <stdio.h>
#include <string.h>
int main() {
    char szoveg1[] = "Hello, ";
    char szoveg2[] = "world!";
    char eredmeny[100];  // Előre rögzített méretű karaktertömb
    strcpy(eredmeny, szoveg1);
    strcat(eredmeny, szoveg2);
    printf("Az összefűzött karaktertömb: %s\n", eredmeny);
}
```

4. Karaktertömb megfordítása A feladat: Írj egy programot, amely egy karaktertömböt megfordít, majd kiírja az eredményt!

```
#include <stdio.h>
#include <string.h>
int main() {
    char szoveg[] = "Hello, world!";
```

```
    int hossz = strlen(szoveg);
    char megforditott[100];
    for (int i = 0; i < hossz; i++) {
        megforditott[i] = szoveg[hossz - 1 - i];
    }
    megforditott[hossz] = '\0';
    printf("A megfordított karaktertömb: %s\n", megforditott);
}
```

5. Karaktertömbben szóközök eltávolítása A feladat: Írj egy programot, amely egy karaktertömbből eltávolítja az összes szóközt, majd kiírja az eredményt!

```
#include <stdio.h>
#include <string.h>
int main() {
    char szoveg[] = "Ez egy példa mondat szóközökkel.";
    char eredmeny[100]; // Előre rögzített méretű karaktertömb
    int j = 0;
    for (int i = 0; szoveg[i] != '\0'; i++) {
        if (szoveg[i] != ' ') {
            eredmeny[j] = szoveg[i];
            j++;
        }
    }
    eredmeny[j] = '\0';
    printf("A szóközök nélküli karaktertömb: %s\n", eredmeny);
}
```

6. Karakterek számolása egy karaktertömbben rekurzióval A feladat: Írj egy programot, amely rekurzív függvény segítségével számolja meg, hány 'e' karakter található egy adott karaktertömbben!

7. Karaktertömb szavakra bontása A feladat: Írj egy programot, amely egy karaktertömbben lévő szavakat bontja fel, majd kiírja az egyes szavakat!

```
#include <stdio.h>
#include <string.h>
int main() {
    char szoveg[] = "Ez egy példa mondat szóközökkel.";
    char* token = strtok(szoveg, " ");
    while (token != NULL) {
        printf("Szó: %s\n", token);
        token = strtok(NULL, " ");
    }
}
```

8. Karakterek összehasonlítása A feladat: Írj egy programot, amely összehasonlítja két karaktertömb tartalmát, és kiírja, hogy azonosak-e!

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

```
#include <string.h>
int main() {
    char szoveg1[] = "Hello, world!";
    char szoveg2[] = "Hello, universe!";
    if (strcmp(szoveg1, szoveg2) == 0) {
        printf("A két karaktertömb azonos.\n");
    } else {
        printf("A két karaktertömb különbözik.\n");
    }
    return 0;
}
```

9. Karakterek cseréje egy karaktertömbben A feladat: Írj egy programot, amely egy karaktertömbben megcseréli az összes 'a' karaktert 'x' karakterre!

```
#include <stdio.h>
#include <string.h>
int main() {
    char szoveg[] = "A karaktertömb cseréje a programozásban.";
    for (int i = 0; szoveg[i] != '\0'; i++) {
        if (szoveg[i] == 'a') {
            szoveg[i] = 'x';
        }
    }
    printf("Az 'a' karakterek cseréje után: %s\n", szoveg);
}
```

10. Karaktertömb összehasonlítása egy mintaszöveggel A feladat: Írj egy programot, amely egy karaktertömbben keres egy adott mintaszöveget, majd kiírja, hogy megtalálható-e benne!

```
#include <stdio.h>
#include <string.h>
int main() {
    char szoveg[] = "Ez egy példa mondat szóközökkel.";
    char mintaszoveg[] = "példa";
    if (strstr(szoveg, mintaszoveg) != NULL) {
        printf("A mintaszöveg megtalálható a karaktertömbben.\n");
    } else {
        printf("A mintaszöveg nem található a karaktertömbben.\n");
    }
}
```

Komplex feladatok: Két mátrix összeszorzása

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 100
// Függvény a mátrix kiírásához
void printMatrix(int matrix[MAX][MAX], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
```

```
        for (int j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
}
```

Függvény a két mátrix szorzatának kiszámításához <sxh c> void multiplyMatrices(int matrix1[MAX][MAX], int rows1, int cols1, int matrix2[MAX][MAX], int rows2, int cols2, int result[MAX][MAX]) { Ellenőrzés: a két mátrix szorzatának megfelelő mérete

```
  if (cols1 != rows2) {
      printf("A ket matrix szorzatanak merete nem megfelelo.\n");
      exit(1); // Kilépés hibakóddal
  }
```

```
  for (int i = 0; i < rows1; i++) {
      for (int j = 0; j < cols2; j++) {
          result[i][j] = 0;
```

```
          for (int k = 0; k < cols1; k++) {
              result[i][j] += matrix1[i][k] * matrix2[k][j];
          }
      }
  }
```

} </sxh> int main() {

```
  // Véletlenszám inicializálása
  srand(time(NULL));
```

```
  // Mátrix méreteinek bekérése
  int rows1, cols1, rows2, cols2;
```

```
  printf("Adja meg az elso matrix meretet (sor oszlop): ");
  scanf("%d %d", &rows1, &cols1); //C99 előtt két sorba kell írni, két scanf
(codeblock)
```

```
  printf("Adja meg a masodik matrix meretet (sor oszlop): ");
  scanf("%d %d", &rows2, &cols2); //C99 előtt két sorba kell írni, két scanf
(codeblock)
```

```
  if (rows1 > MAX || cols1 > MAX || rows2 > MAX || cols2 > MAX) {
      printf("A megadott méret túl nagy. Maximum méret: %d\n", MAX);
      return 1; // Kilépés hibakóddal
  }
```

```
  // Mátrixok létrehozása és inicializálása véletlenszámokkal
  int matrix1[MAX][MAX], matrix2[MAX][MAX], result[MAX][MAX];
```

```
  for (int i = 0; i < rows1; i++) {
      for (int j = 0; j < cols1; j++) {
          matrix1[i][j] = rand() % 100; // Véletlenszám 0 és 99 között
      }
  }
```

```
  for (int i = 0; i < rows2; i++) {
      for (int j = 0; j < cols2; j++) {
          matrix2[i][j] = rand() % 100; // Véletlenszám 0 és 99 között
      }
  }
```

```
  // Eredeti mátrixok kiírása
  printf("Eredeti elso matrix:\n");
  printMatrix(matrix1, rows1, cols1);
  printf("\nEredeti masodik matrix:\n");
  printMatrix(matrix2, rows2, cols2);
```

```
  // Mátrixok szorzata
  multiplyMatrices(matrix1, rows1, cols1, matrix2, rows2, cols2, result);
```

```
  // Szorzat kiírása
  printf("\nA ket matrix szorzata:\n");
  printMatrix(result, rows1, cols2);
```

```
  return 0; // Sikeres végrehajtás
```

} </sxh> Mátrix maximális részmátrix megtalálása

```
#include <stdio.h>
#include <stdlib.h>
#define MERET 100
int matrix[MERET + 1][MERET + 1];
int main() {
    int n, bal, felso, jobb, also, i, j;
    long max;
    scanf("%d", &n);

    for (i = 0; i < n; i++){
        for (j = 0; j < n; j++){
            scanf("%d", &matrix[i][j]);
        }
    }
    max = matrix[0][0];
    for (bal = 0; bal < n; bal++){
        for (felso = 0; felso < n; felso++){
            for (jobb = bal; jobb < n; jobb++){
                for (also = felso; also < n; also++) {
                    long osszeg = 0;
```

```
                    for (i = bal; i <= jobb; i++){
                        for (j = felso; j <= also; j++){
                            osszeg += matrix[i][j];
                        }
                    }
                    if (osszeg > max){
                        max = osszeg;
                    }
                }
            }
        }
    }
    printf("%ld\n", max);
}
```