Programarea Calculatoarelor
Exemplu de sarcină rezolvată pentru Lucrarea de laborator 5

Sarcina: Să se determine produsul elementelor desupra diagonalei secundare și să se înlocuiască elementele de pe diagonala principală cu zero.

Exemplu 1 de cod cu matrice alocată într-un singur bloc de memorie continuu, matrice simulată ca un vector de lungime n * n.

Program fără funcții:

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

int main() {
    int n, i, j;
    int *tab;
    int produs = 1;   // produsul elementelor deasupra diagonalei secundare

    // Validarea valorii n
    do {
        printf("Introduceti n (o valoare >= 2): ");
        printf("n = "); scanf("%d", &n);
        if (n < 2) printf("Valoare invalida!\n");
    } while (n < 2);

    // Alocare dinamica pentru o matrice n x n intr-un singur bloc
    tab = (int *)malloc(n * n * sizeof(int));
    if (tab == NULL) {
        printf("Eroare la alocarea memoriei.\n");
        return 1;
    }

    // Citirea elementelor matricei
    printf("Introduceti elementele matricei:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", (tab + i * n + j));
        }
    }

    // Calculul produsului elementelor deasupra diagonalei secundare
    // Diagonala secundara are indicii i + j == n - 1
    // Deasupra diagonalei secundare: i + j < n - 1
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (i + j < n - 1) {
                produs *= *(tab + i * n + j);
            }
        }
    }
 printf("Produsul elementelor deasupra diagonalei secundare este: %d\n", produs);
```

```
    // Afisarea matricei pana la modificare
    printf("Matricea initiala:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", *(tab + i * n + j));
        }
        printf("\n");
    }

    // Inlocuirea elementelor de pe diagonala principala cu 0
    // Diagonala principala: i == j, deci vom folosi tab[i][i]
    for (i = 0; i < n; i++) {
        *(tab + i * n + i) = 0;
    }

    // Afisarea matricei modificate
    printf("Matricea modificata:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", *(tab + i * n + j));
        }
        printf("\n");
    }

    // Eliberarea memoriei
    free(tab);

    return 0;
}
```

Program cu funcții:

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

int* alocareMatrice(int n) {
    int *tab = (int *)malloc(n * n * sizeof(int));
    if (tab == NULL) {
        printf("Eroare la alocarea memoriei.\n");
        exit(1);
    }
    return tab;
}

void citireMatrice(int *tab, int n) {
    printf("Introduceti elementele matricei:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", (tab + i * n + j));
        }
    }
}
```

```
void afisareMatrice(int *tab, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", *(tab + i * n + j));
        }
        printf("\n");
    }
}
```

```
int produsDeasupraDiagSecundara(int *tab, int n) {
    int produs = 1;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i + j < n - 1) {     // deasupra diag. secundare
                produs *= *(tab + i * n + j);
            }
        }
    }
    return produs;
}
```

```
void inlocuireDiagonalaPrincipalaCuZero(int *tab, int n) {
    for (int i = 0; i < n; i++) {
        *(tab + i * n + i) = 0;
    }
}
```

```
int main() {
    int n;
    // Validarea valorii n
    do {
        printf("Introduceti n (o valoare >= 2): ");
        printf("n = "); scanf("%d", &n);
        if (n < 2) printf("Valoare invalida!\n");
    } while (n < 2);
```

```
    int *tab = alocareMatrice(n);
    citireMatrice(tab, n);
    printf("Matricea initiala:\n");
    afisareMatrice(tab, n);
    int produs = produsDeasupraDiagSecundara(tab, n);
    printf("Produsul elementelor deasupra d. secundare este: %d\n", produs);
    inlocuireDiagonalaPrincipalaCuZero(tab, n);
    printf("Matricea modificata:\n");
    afisareMatrice(tab, n);
```

```
    free(tab);
    return 0;
}
```

Exemplu 2 de cod cu matrice alocată separat pentru fiecare linie. Se alocă un vector de pointeri către n linii.
Program fără funcții:

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

int main() {
    int n, i, j;
    int produs = 1;
    // Validarea valorii n
    do {
        printf("Introduceti n (o valoare >= 2): ");
        printf("n = "); scanf("%d", &n);
        if (n < 2) printf("Valoare invalida!\n");
    } while (n < 2);

    // Alocare dinamica a vectorului de pointeri
    int **arr = (int **)malloc(n * sizeof(int *));
    if (arr == NULL) {
        printf("Eroare la alocarea liniilor.\n");
        return 1;
    }

    // Alocare dinamica pentru fiecare linie
    for (i = 0; i < n; i++) {
        arr[i] = (int *)calloc(n, sizeof(int));
        if (arr[i] == NULL) {
            printf("Eroare la alocarea coloanelor.\n");
            return 1;
        }
    }

    // Citirea matricei
    printf("Introduceti elementele matricei:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", *(arr + i) + j);
        }
    }

    // Calculul produsului deasupra diagonalei secundare: i + j < n - 1
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (i + j < n - 1) {
                produs *= *(*(arr + i) + j);
            }
        }
    }
    printf("Produsul elementelor deasupra d. secundare este: %d\n", produs);
```

```
    // Afisarea matricei pana la modificare
    printf("Matricea initiala:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", *(*(arr + i) + j));
        }
        printf("\n");
    }
    // Inlocuirea elementelor de pe diagonala principala cu 0
    for (i = 0; i < n; i++) {
        *(*(arr + i) + i) = 0;
    }
    // Afisarea matricei modificate
    printf("Matricea actualizata:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", *(*(arr + i) + j));
        }
        printf("\n");
    }
    // Eliberarea memoriei
    for (i = 0; i < n; i++)
        free(arr[i]);
    free(arr);
    return 0;
}
```

Program cu funcții:

```
#include <stdio.h>
#include <stdlib.h>
int** alocareMatrice(int n) {
    int **arr = (int **)malloc(n * sizeof(int *));
    if (arr == NULL) {
        printf("Eroare la alocarea vectorului de pointeri.\n");
        exit(1);
    }

    for (int i = 0; i < n; i++) {
        arr[i] = (int *)calloc(n, sizeof(int));
        if (arr[i] == NULL) {
            printf("Eroare la alocarea liniei %d.\n", i);
            exit(1);
        }
    }
    return arr;
}
void citireMatrice(int **arr, int n) {
    printf("Introduceti elementele matricei:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", (*(arr + i) + j));
}
```

```c
void afisareMatrice(int **arr, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            printf("%d ", *(*(arr + i) + j));
        printf("\n");
    }
}
```

```c
int produsDeasupraDiagSecundara(int **arr, int n) {
    int produs = 1;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (i + j < n - 1)
                produs *= *(*(arr + i) + j);
    return produs;
}
```

```c
void inlocuireDiagPrincipalaCuZero(int **arr, int n) {
    for (int i = 0; i < n; i++)
        *(*(arr + i) + i) = 0;
}
```

```c
void eliberareMatrice(int **arr, int n) {
    for (int i = 0; i < n; i++)
        free(arr[i]);
    free(arr);
}
```

```c
int main() {
    int n;
    // Validarea valorii n
    do {
        printf("Introduceti n (o valoare >= 2): ");
        printf("n = "); scanf("%d", &n);
        if (n < 2) printf("Valoare invalida!\n");
    } while (n < 2);

    int **arr = alocareMatrice(n);
    citireMatrice(arr, n);
    printf("Matricea initiala:\n");
    afisareMatrice(arr, n);
    int produs = produsDeasupraDiagSecundara(arr, n);
    printf("Produsul elementelor deasupra d. secundare este: %d\n", produs);
    inlocuireDiagPrincipalaCuZero(arr, n);
    printf("Matricea actualizata:\n");
    afisareMatrice(arr, n);
    eliberareMatrice(arr, n);

    return 0;
}
```