20. Динамические матрицы.
План ответа:
Способы выделения памяти для динамических матриц: идеи, реализации, анализ преимуществ и недостатков.

способы представления:

1. одномерный массив
Как одномерный массив; представление как статического массива.
Выделение и освобождение памяти тривиально.

Достоинства
Простота выделения и освобождения памяти
Минимум занимаемого места
Возможность использовать как одномерный массив
Недостатки
Необходимость всё время обращаться к элементам по сложному индексу
Средства для контроля работы с памятью не могут отследить выход за пределы строки

```
double *data;
int n = 3, m = 2;

data = malloc(n * m * sizeof(double));
if (data)
{
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            // Обращение к элементу i, j
            data[i*m+j] = 0.0;

    free(data);
}
```

2. указатели на строки
Объявляется массив указателей на массивы-строки, элементы которых имеют некий тип, и т.д.
Память под массив строк и массивы-строки выделяется и освобождается отдельно.

Алгоритм выделения памяти
Вход: количество строк (n) и количество столбцов (m)
Выход: указатель на массив строк матрицы (p)

Выделить память под массив указателей (p)
Обработать ошибку выделения памяти
В цикле по количеству строк матрицы (0 <= i < n)
— Выделить память под i-ую строку матрицы (q)
— Обработать ошибку выделения памяти
— p[i]=q

Алгоритм освобождения памяти
Вход: указатель на массив строк матрицы (p) и количество строк (n)

В цикле по количеству строк матрицы (0 <= i < n)
— Освободить память из-под i-ой строки матрицы
Освободить память из-под массива указателей (p)

Достоинства
Возможность обмена строки через обмен указателей.
СРПР может отследить выход за пределы строки.
Недостатки
Сложность выделения и освобождения памяти.
Память под матрицу "не лежит" одним куском.

```
void free_matrix_rows(double **data, int n)
{
    for (int i = 0; i < n; i++)
       // free можно передать NULL
       free(data[i]);

    free(data);
}


double** allocate_matrix_rows(int n, int m)
{
    double **data = calloc(n, sizeof(double*));
    if (!data)
        return NULL;

    for (int i = 0; i < n; i++)
    {
        data[i] = malloc(m * sizeof(double));
    if (!data[i])
    {
        free_matrix_rows(data, n);
        return NULL;
        }
    }
    return data;
}
```

3. одним дампом

Выделяется единый дамп памяти, первая часть которого идёт на массивы указателей на строки, вторая содержит значения.

Алгоритм выделения памяти

Вход: количество строк (n) и количество столбцов (m)

Выход: указатель на массив строк матрицы (p)

Выделить память под массив указателей на строки и элементы матрицы (p)

Обработать ошибку выделения памяти

В цикле по количеству строк матрицы (0 <= i < n)

– Вычислить адрес i-ой строки матрицы (q)

– p[i]=q

Достоинства

Простота выделения и освобождения памяти.

Возможность использовать как одномерный массив.

Перестановка строк через обмен указателей.

Недостатки

Сложность начальной инициализации.

СРПР не может отследить выход за пределы строки.

```
double** allocate_matrix_solid(int n, int m)
{
    double **data = malloc(n * sizeof(double*) +
                                n * m * sizeof(double));
    if (!data)
        return NULL;

    for (int i = 0; i < n; i++)
        data[i] = (double*)((char*) data +
                                                n * sizeof(double*)
 +
                                           i * m * sizeof(double));
    return data;
}
```