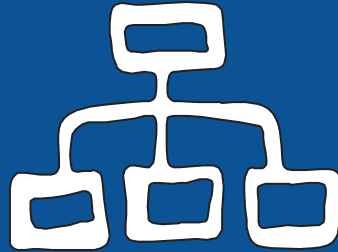


Алгоритмы и Алгоритмические Языки

Семинар #12:

1. Перечисления в языке C и коды ошибок.
2. Организация двумерного массива.
3. Работа с текстовыми файлами.

Перечисления в языке С и коды ошибок



Массив с проверкой доступа

```
// Представление вектора в N-мерном пространстве
struct Vector
{
    // Элементы вектора
    double* data;
    // Размерность пространства (кол-во элементов)
    size_t size;
};
```

Возможный API (программный интерфейс) массива:

```
struct Vector vector_alloc(size_t size);
void          vector_free (struct Vector* vec);
double        vector_get  (const struct Vector* vec, size_t index);
void          vector_set  (struct Vector* vec, size_t index, double element);
```

Какие проблемы могут возникнуть в каждой из операций?

Перечисления и коды ошибок

```
// Тип данных, задающий код возврата
typedef enum
{
    // Операция выполнена без ошибок
    RET_OK      = 0,
    // Операция не выполнена, т.к. для её выполнения не хватает памяти
    RET_NOMEM   = 1,
    // Операция не выполнена, т.к. аргументы операции некорректны
    RET_INVALID = 2,
    // Операция не выполнена, т.к. операция с файлом вернула ошибку
    RET_FILEIO  = 3
} RetCode;
```

```
RetCode vector_alloc(struct Vector* vec, size_t size);
RetCode vector_free (struct Vector* vec);
RetCode vector_get  (const struct Vector* vec, size_t index, double* element);
RetCode vector_set  (struct Vector* vec, size_t index, double element);
```

Какие коды ошибок могут возникнуть в каждой из операций?

Применение перечислений

```
RetCode vector_alloc(struct Vector* vec, size_t size)
{
    if (vec == NULL)
    {
        return RET_INVALID;
    }

    double* data = (double*) calloc(size, sizeof(double));
    if (data == NULL)
    {
        return RET_NOMEM;
    }
}
```

```
// Аллоцируем неинициализированный вектор
RetCode ret = vector_alloc(&vec, SIZE);
verify_contract(ret == RET_OK,
    "Unable to allocate vector: %s\n",
    retcode_str(ret));
```

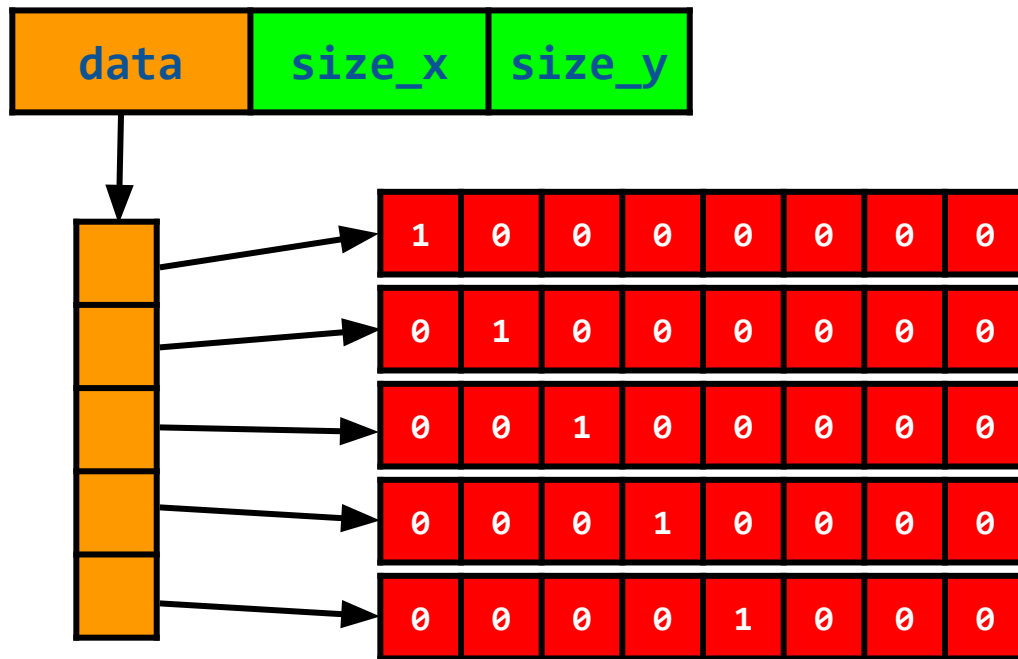
```
const char* retcode_str(RetCode code)
{
    switch (code)
    {
        case RET_OK:
        {
            return "everything is ok";
        }
        case RET_NOMEM:
        {
            return "not enough memory";
        }
        case RET_INVALID:
        {
            return "invalid argument";
        }
        case RET_FILEIO:
        {
            return "failed file operation";
        }
        default:
        {
            return "invalid return code";
        }
    }
}
```

Организация двумерного массива

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Структура данных “Матрица”

Размещение матрицы в памяти:



Как ещё можно разместить в памяти двумерный массив?

Аллокация двумерного массива

```
RetCode matrix_alloc(struct Matrix* mat, size_t size_x, size_t size_y)
{
    if (mat == NULL)
    {
        return RET_INVALID;
    }

    // Выделяем массив, содержащий строки матрицы
    double** data = (double**) calloc(size_y, sizeof(double*));
    if (data == NULL)
    {
        return RET_NOMEM;
    }

    // Выделяем строки матрицы
    for (size_t y = 0; y < size_y; ++y)
    {
        data[y] = (double*) calloc(size_x, sizeof(double));
        if (data[y] == NULL)
        {
            return RET_NOMEM;
        }
    }
}
```

В чём отличие **malloc** от **calloc**? Какие значения будут в матрице?

Освобождение двумерного массива

```
RetCode matrix_free(struct Matrix* mat)
{
    if (mat == NULL || mat->data == NULL)
    {
        return RET_INVALID;
    }

    // Освобождаем строки матрицы
    for (size_t y = 0; y < mat->size_y; ++y)
    {
        if (mat->data[y] == NULL)
        {
            return RET_INVALID;
        }

        free(mat->data[y]);
    }

    // Освобождаем массив строк
    free(mat->data);

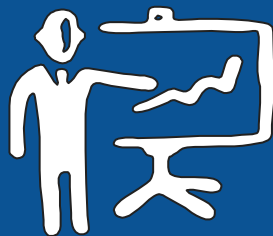
    // Зануляем указатель, чтобы защититься от повторного вызова matrix_free.
    mat->data = NULL;

    return RET_OK;
}
```

Работа с текстовыми файлами



Вопросы?



Красивые иконки взяты с сайта handdrawngoods.com