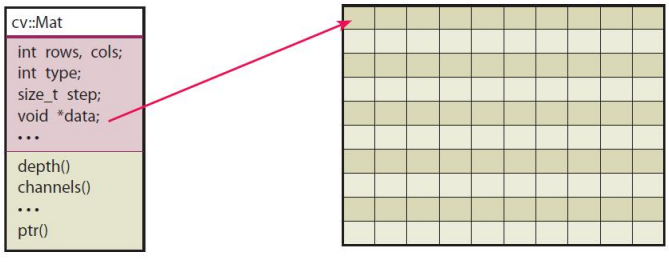
3장 OPENCV의 기초

1

Mat 클래스

- Mat 클래스는 OpenCV에서 영상을 담을 때 사용하는 데이터 구조로서 OpenCV 라이브러리의 핵심 요소



2

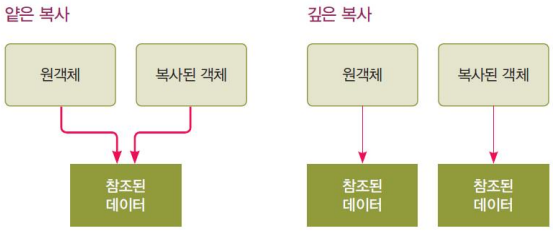
헤더와 데이터 분리

```
void sub() {
        Mat A;                          // ① 여기서는 헤더만 생성된다.
        A = imread("lenna.jpg", IMREAD_COLOR); // ② 여기서 동적 메모리가 할당된다.
        ...
        // ③ 함수가 종료되면 자동적으로 동적 메모리가 해제된다.
}
```

3

얕은 복사 사용

```
void sub() {
        Mat A;                          // ① 여기서는 헤더만 생성된다.
        A = imread("lenna.jpg", IMREAD_COLOR); // ② 여기서 동적 메모리가 할당된다.
        ...
        // ③ 함수가 종료되면 자동적으로 동적 메모리가 해제된다.
}
```



4

Mat 객체의 속성 출력

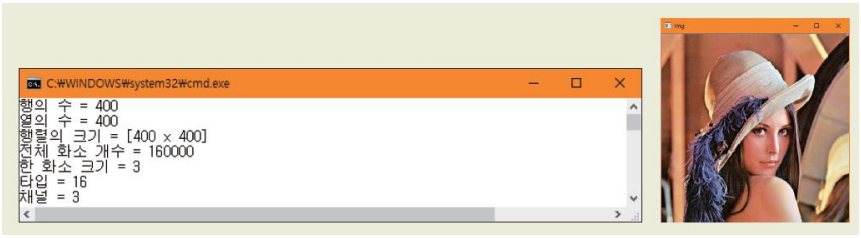
```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace cv;
using namespace std;

int main()
{
        Mat img = imread("d:/lenna.jpg");
        if (img.empty()) { cout << "영상을 읽을 수 없음" << endl;        return -1; }
        imshow("img", img);

        cout << "행의 수 = " << img.rows << endl;
        cout << "열의 수 = " << img.cols << endl;
        cout << "행렬의 크기 = " << img.size() << endl;
        cout << "전체 화소 개수 = " << img.total() << endl;
        cout << "한 화소 크기 = " << img.elemSize() << endl;
        cout << "타입 = " << img.type() << endl;
        cout << "채널 = " << img.channels() << endl;
        waitKey(0);
        return 0;
}
```

5

실행 결과



6

화소 데이터 저장 방법(그레이스케일 영상)

	0열	1열	...	m-1열
0행	(0, 0)	(0, 1)	...	(0, m-1)
1행	(1, 0)	(1, 1)	...	(1, m-1)
...	...	...	...	...
n-1행	(n-1, 0)	(n-1, 1)	...	(n-1, m-1)

7

화소 데이터 저장 방법(컬러 영상)

8

화소값 출력하기

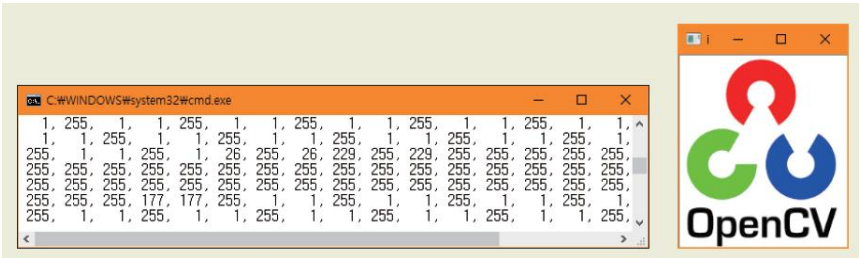
```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace cv;
using namespace std;

int main()
{
        Mat img = imread("d:/opencv.png");
        if (img.empty()) { cout << "영상을 읽을 수 없음" << endl;        return -1; }
        imshow("img", img);

        cout << img << endl;
        waitKey(0);
        return 0;
}
```

9

실행 결과



10

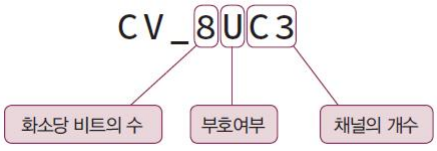
명시적으로 Mat 객체 만들기

```
Mat (int rows, int cols, int type, const Scalar &s);
```

생성자	설명
rows	영상의 행 개수
cols	영상의 열 개수
type	화소의 값을 저장하는 데 사용되는 자료형
s	화소의 초기값

11

화소의 자료형 지정




```
Mat A(3, 3, CV_32F);              // 1채널의 3x3 실수(32비트) 행렬
Mat B(10, 1, CV_64FC2);           // 2채널의 10x1 실수(64비트) 행렬
Mat C(Size(1920, 1080), CV_8UC3); // 1080행과 1920열을 가지는 3채널 영상
```

12

화소의 자료형 지정

자료형	비트수	설명
CV_8U	8	8-bit unsigned integer: uchar (0..255)
CV_8S	8	8-bit signed integer: schar (-128..127)
CV_16U	16	16-bit unsigned integer: ushort (0..65535)
CV_16S	16	16-bit signed integer: short (-32768..32767)
CV_32S	32	3-bit signed integer: int (-2147483648..2147483647)
CV_32F	32	32-bit floating-point number: float (-FLT_MAX..FLT_MAX, INF, NAN)
CV_64F	64	64-bit floating-point number: double (-DBL_MAX..DBL_MAX, INF, NAN)

13

화소 초기화

- cv::Scalar(0, 0, 255)

14

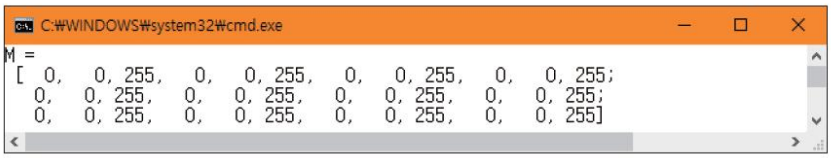
예제

```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace cv;
using namespace std;

int main()
{
        Mat M(3, 4, CV_8UC3, Scalar(0, 0, 255));
        cout << "M = " << endl << " " << M << endl << endl;
        return 0;
}
```

15

실행 결과



16

Lab: 명시적으로 만든 Mat 객체를 표시

```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace cv;
using namespace std;

int main()
{
        Mat M(600, 800, CV_8UC3, Scalar(0, 255, 0));
        if (M.empty()) { cout << "영상을 읽을 수 없음" << endl;        return -1; }
        imshow("img", M);

        waitKey(0);
        return 0;
}
```

17

실행 결과

18

zeros(), ones(), eye()를 사용하여 행렬 생성하기

```
...
int main()
{
        Mat E = Mat::eye(4, 4, CV_64F);
        cout << "E = " << endl << " " << E << endl << endl;
        Mat O = Mat::ones(2, 2, CV_32F);
        cout << "O = " << endl << " " << O << endl << endl;
        Mat Z = Mat::zeros(3, 3, CV_8UC1);
        cout << "Z = " << endl << " " << Z << endl << endl;
        return 0;
}
```

19

실행 결과

20

Q & A