WeChat Teaching Assistant

CS111 - C Programming
MT733

```c
int main()
{
    int i;
    for(i=1; i<6; i++)
    {
        switch(i)
        {
        case 2:
            printf("%d ", i);
            break;
        default:
            continue;
        }
        printf("%d ", i);
    }
    printf("%d\n", i);
    return(0);
}
```

/* Quiz 1 */

Quiz 2

- Which of the following C statements is/are incorrect?

A. int a[6] = {0, 2, 4, 6, 8};

B. int a = {0, 2, 4, 6, 8};

C. int a[] = {[5] = 4, [3] = 8};

D. int a[4] = {0, 2, 4, 6, 8};

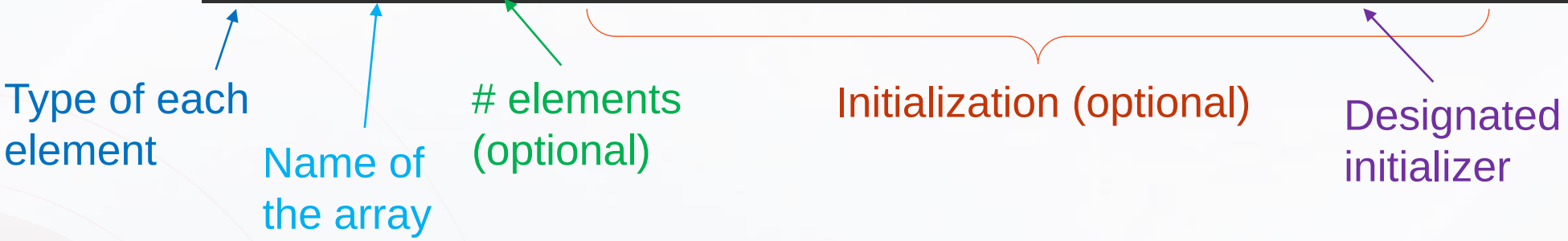
E. int a[3] = 4;

What have we learned?

Arrays

- Declaration & initialization

```
int arr1[10] = {5, 1, [4] = 3, 7, 2, [8] = 6};
```



- Indexing

Index starts from 0 in C!

```
sum = 0;
for (i = 0; i < N; i++)
    sum += arr1[i];      /* sums all elements */
```

Example: Checking repeated digits

- Write a program to check whether any of the digits in a number appears more than once.

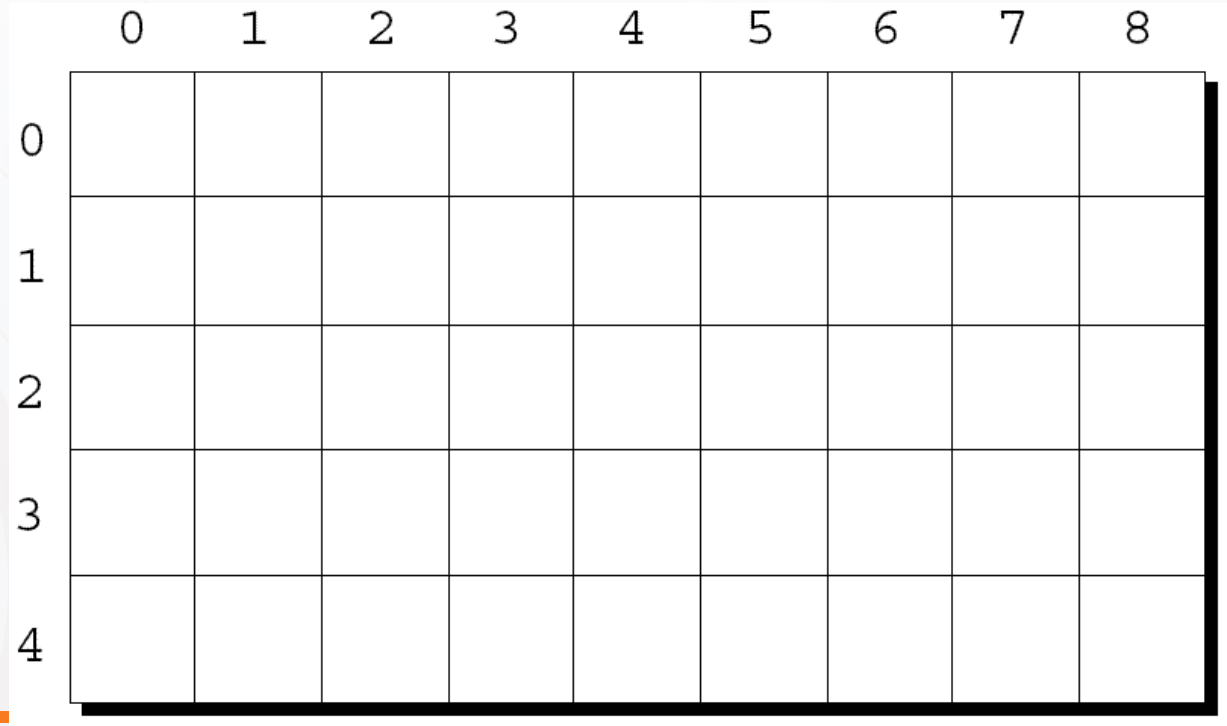
```
Enter a number: 28212
    Repeated digit

Enter a number: 935742
    No repeated digit
```

```c
int main()
{
    int digit_seen[10] = {0};
    int digit;
    long n;
    printf("Enter a number: ");
    scanf("%ld", &n);
    while (n > 0)
    {
        digit = n % 10;
        if (digit_seen[digit]) break;
        digit_seen[digit] = 1;
        n /= 10;
    }
    if (n > 0) printf("Repeated digit\n");
    else printf("No repeated digit\n");
    return 0;
}
```

Multidimensional array

```
int m[5][9];
```

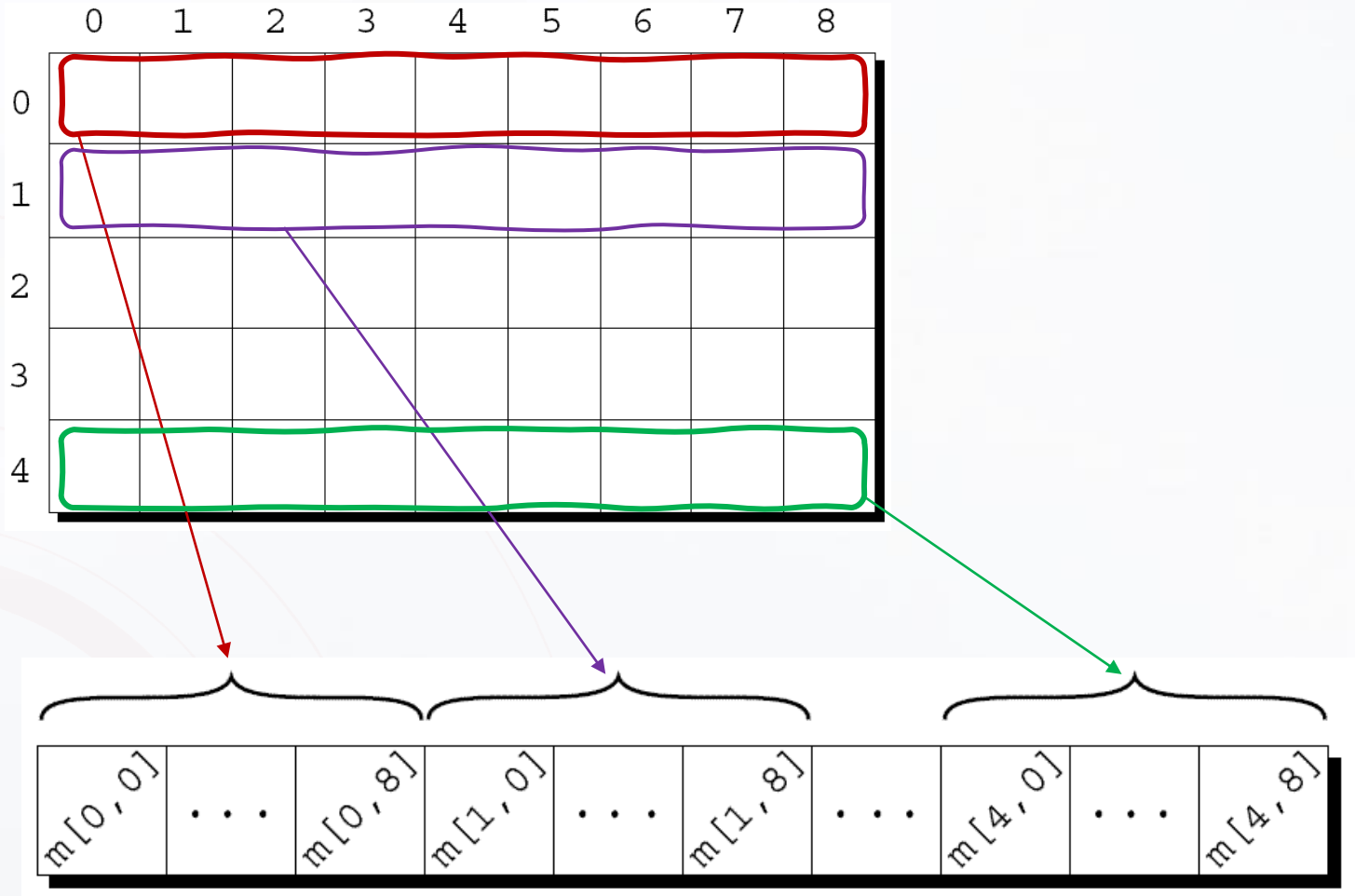
- A two-dimensional array with 5 rows and 9 columns.
- Both rows and columns are indexed from 0:



It's just a matrix!

m[i, j] ✗

Storage of a multidimensional array



row-major order

Accessing multidimensional array elements

```
#define M 12
#define N 10

    double ident[M][N];
    int i, j;
    for (i = 0; i < M; i++)
    {
        for (j = 0; j < N; j++)
        {
            if (i == j)
                ident[i][j] = 1.0;
            else
                ident[i][j] = 0.0;
        }
    }
```

Nested for loops are ideal for accessing multidimensional array elements sequentially

Initialization

```
int m[5][9] = {{1, 1, 1, 1, 1, 0, 1, 1, 1},
               {0, 1, 0, 1, 0, 1, 0, 1, 0},
               {0, 1, 0, 1, 1, 0, 0, 1, 0},
               {1, 1, 0, 1, 0, 0, 0, 1, 0},
               {1, 1, 0, 1, 0, 0, 1, 1, 1}};
```

```
int m[5][9] = {{1, 1, 1, 1, 1, 0, 1, 1, 1},
               {0, 1, 0, 1, 0, 1, 0, 1},
               {0, 1, 0, 1, 1, 0, 0, 1},
               {1, 1, 0, 1, 0, 0, 0, 1},
               {1, 1, 0, 1, 0, 0, 1, 1, 1}};
```

Omitted elements/rows are filled with 0s

```
int m[5][9] = {{1, 1, 1, 1, 1, 0, 1, 1, 1},
               {0, 1, 0, 1, 0, 1, 0, 1, 0},
               {0, 1, 0, 1, 1, 0, 0, 1, 0}};
```

Initialization

- Inner braces may be neglected (but *unrecommended*)

```
int m[5][9] = {1, 1, 1, 1, 1, 0, 1, 1, 1,
               0, 1, 0, 1, 0, 1, 0, 1, 0,
               0, 1, 0, 1, 1, 0, 0, 1, 0,
               1, 1, 0, 1, 0, 0, 0, 1, 0,
               1, 1, 0, 1, 0, 0, 1, 1, 1};
```

- Designated initializers

```
double Identity[3][3] = {[0][0] = 1.0, [1][1] = 1.0,
    [2][2] = 1.0};
```

Constant Arrays

- Declaration:

```
const char hex_chars[] =
      {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
      'A', 'B', 'C', 'D', 'E', 'F'};
```

- Advantages:
  - ✓Explicitly tell yourself (and others) that the array is not to be changed;
  - ✓Helps the compiler to catch errors.
- Not just for arrays.

const vs. #define

- Handled by the compiler
- A variable that is not to be changed
- A const variable must have a type
- Ends with a ;

- Handled by the pre-processor
- Just copy & paste
- Does *not* check type
- No ;

Example: Dealing a hand of cards

- The program deals a *random* hand from a standard deck of playing cards.
- Each card in a standard deck has a *suit* (clubs, diamonds, hearts, or spades) and a *rank* (2, 3, …, 10, J, Q, K, A).
- The user will specify number of cards in hand:

```
Enter number of cards in hand: 5
Your hand: 7c 2s 5d as 2h
```

Q1: how to pick cards randomly

- Library functions to use:
  - ✓**`rand`** (from `<stdlib.h>`) – produces an apparently random number each time it's called.
  - ✓**`srand`** (from `<stdlib.h>`) – initializes C's random number generator.
  - ✓**`time`** (from `<time.h>`) – returns the current time, encoded in a single number.

- How to cast the random number onto 52 cards?
  - ✓The `%` (remainder) operator

```c
#include <stdlib.h>
#include <time.h>

#define NUM_SUITS 4
#define NUM_RANKS 13

int main(void)
{
    int num_cards, rank, suit;
    const char rank_code[] = {'A','2','3','4','5','6','7',
                              '8','9','X','J','Q','K'};
    const char suit_code[] = {'C','D','H','S’};
    …
    srand((unsigned) time(NULL));
    suit = rand() % NUM_SUITS;     /* picks a random suit */
    rank = rand() % NUM_RANKS;     /* picks a random rank */
    …
}
```

Q2: How do we avoid picking the same card twice?

- We use an array `in_hand` to record the *status* of each card (taken or not).
- The array has 4 rows (suits) and 13 columns (ranks).
- All elements of the array will be 0 (false) to start with.
- Each time we pick a card at random, we'll check whether the corresponding element of `in_hand` is true or false.
  - If it's true, we'll have to pick another card.
  - If it's false, we'll store 1 (true) in that element to remind us later that this card has already been picked.

```
int in_hand[NUM_SUITS][NUM_RANKS] = {0};

srand((unsigned) time(NULL));
while (num_cards > 0)
{
    suit = rand() % NUM_SUITS;     /* picks a random suit */
    rank = rand() % NUM_RANKS;     /* picks a random rank */
    if (!in_hand[suit][rank])
    {
        in_hand[suit][rank] = 1;
        num_cards--;
        printf(" %c%c", rank_code[rank], suit_code[suit]);
    }
}
```