Fast campus

컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 13강 – 다차원 배열과 포인터 배열

Fast campus

학습 목표

다차원 배열과 포인터 배열

1) 1차원 배열을 넘어 다양한 차원의 배열을 표현하는 방법을 이해합니다.
2) 포인터를 이용해 2차원 배열을 다루는 방법을 학습합니다.

다차원 배열과 포인터 배열

2차원 배열의 필요성

1) 2차원 배열은 굉장히 많은 목적으로 사용됩니다.
2) 행렬 데이터를 표현할 때, 그래프 알고리즘을 처리할 때, 다수의 실생활 데이터를 처리할 때 등입니다.
3) 흔히 우리가 보는 표 구조가 2차원 배열과 흡사합니다.

이름	영어 성적	수학 성적	국어 성적
홍길동	85	97	79
유관순	100	89	98
이순신	99	77	99
장보고	89	70	78
신립	95	98	98

Fast campus

다차원 배열과 포인터 배열

2차원 배열의 초기화

1) 2차원 배열은 1차원 배열이 중첩되었다는 의미로 [](대괄호)를 두 번 연속하여 씁니다.

자료형 배열이름[행][열] = { {값, 값, 값, ...}, {값, 값, 값, ...}, ... }

```
int a[10][10];
```

다차원 배열과 포인터 배열

2차원 배열의 초기화

1) 2차원 배열 또한 기본적으로 0 인덱스부터 시작합니다.

A[0][0]	A[0][1]	A[0][2]
A[1][0]	A[1][1]	A[1][2]
A[2][0]	A[2][1]	A[2][2]
A[3][0]	A[3][1]	A[3][2]
A[4][0]	A[4][1]	A[4][2]

Fast campus

다차원 배열과 포인터 배열

2차원 배열의 사용

1) 2차원 배열은 2중 FOR문과 함께 많이 사용됩니다.

```
#include <stdio.h>

int a[3][3] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

int main(void) {
  int i, j;
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      printf("%d ", a[i][j]);
    }
    printf("₩n");
  }
  system("pause");
  return 0;
}
```

다차원 배열과 포인터 배열

다차원 배열

1) 2차원 배열 이상의 다차원 배열 또한 사용할 수 있습니다.
2) 우리 컴퓨터는 기본적으로 화면에 2차원 형태만 출력할 수 있습니다.

다차원 배열과 포인터 배열

3차원 배열 다루기

```
#include <stdio.h>

int a[2][3][3] = { { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } },
                   { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } } };

int main(void) {
  int i, j, k;
  for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
      for (k = 0; k < 3; k++) {
        printf("%d ", a[i][j][k]);
      }
      printf("₩n");
    }
    printf("₩n");
  }
  system("pause");
  return 0;
}
```

다차원 배열과 포인터 배열

포인터 배열의 구조 분석

1) 배열은 포인터와 동일한 방식으로 동작합니다.
2) 배열의 이름은 배열의 원소의 첫 번째 주소가 됩니다.
3) 유일한 차이점이라고 하면, 포인터는 변수이며 배열의 이름은 상수입니다.

Fast campus

다차원 배열과 포인터 배열

포인터 배열의 구조 분석

1) 배열의 이름이 변수인지 상수인지는 다음의 소스코드를 통해 확인할 수 있습니다.

```
#include <stdio.h>

int main(void) {
  int a = 10;
  int b[10];
  b = &a;
  system("pause");
  return 0;
}
```

Fast campus

다차원 배열과 포인터 배열

포인터 배열의 구조 분석

1) 반대로 포인터를 배열처럼 사용할 수도 있습니다.

```
#include <stdio.h>

int main(void) {
  int a[5] = { 1, 2, 3, 4, 5 };
  int *b = a;
  printf("%d₩n", b[2]);
  system("pause");
  return 0;
}
```

Fast campus

다차원 배열과 포인터 배열

포인터 배열의 구조 분석

1) 배열의 이름은 배열의 첫 번째 원소의 주소라는 것을 기억합니다.

```
#include <stdio.h>

int main(void) {
  int a[5] = { 1, 2, 3, 4, 5 };
  int *b = &a[0];
  printf("%d₩n", b[2]);
  system("pause");
  return 0;
}
```

다차원 배열과 포인터 배열

포인터 배열의 구조 분석

1) 포인터는 연산을 통해 자료형의 크기만큼 이동합니다.
2) 따라서 정수(int)형 포인터는 4바이트(Bytes)씩 이동합니다.

```
#include <stdio.h>

int main(void) {
  int a[5] = { 1, 2, 3, 4, 5 };
  int i;
  for (i = 0; i < 5; i++) {
    printf("%d ", a + i);
  }
  system("pause");
  return 0;
}
```

다차원 배열과 포인터 배열

포인터 배열의 구조 분석

1) 크기가 10인 double형 배열을 선언했을 때 배열의 시작 주소가 X 라고 합니다.
2) 이 때 배열의 마지막 원소의 주소는 몇일까요?

```c
#include <stdio.h>

int main(void) {
  double b[10];
  printf("%d %d₩n", b, b + 9);
  system("pause");
  return 0;
}
```

다차원 배열과 포인터 배열

포인터 배열의 구조 분석

1) 배열을 포인터처럼 사용해 각 원소에 접근할 수도 있습니다.

```
#include <stdio.h>

int main(void) {
  int a[5] = { 1, 2, 3, 4, 5 };
  int i;
  for (i = 0; i < 5; i++) {
    printf("%d ", *(a + i));
  }
  system("pause");
  return 0;
}
```

다차원 배열과 포인터 배열

포인터 배열의 구조 분석

1) 다음 프로그램의 결과는 어떻게 될까요?

```
#include <stdio.h>

int main(void) {
  int a[5] = { 1, 2, 3, 4, 5 };
  int *p = a;
  printf("%d₩n", *(p++));
  printf("%d₩n", *(++p));
  printf("%d₩n", *(p + 2));
  system("pause");
  return 0;
}
```

Fast campus

다차원 배열과 포인터 배열

포인터 배열의 구조 분석

1) 2차원 배열 또한 포인터로 처리할 수 있습니다.

```
#include <stdio.h>

int main(void) {
  int a[2][5] = { { 1, 2, 3, 4, 5 },
                  { 6, 7, 8, 9, 10 } };
  int (*p)[5] = a[1];
  int i;
  for (i = 0; i < 5; i++) {
    printf("%d ", p[0][i]);
  }
  system("pause");
  return 0;
}
```

배운 내용 정리하기

다차원 배열과 포인터 배열

1) 컴퓨터에서 2차원 배열 이상을 표현할 수 있습니다.
2) C언어의 배열은 내부적으로 포인터와 동일하므로 포인터 연산으로 배열을 대체할 수 있습니다.