튜터링 1주차

(TUTOR: 성열암)

응용컴퓨터 프로그래밍

TUTORING — https://github.com/developersung13/cbnu-tutoring

C 언어에서 사용되는 주요한 문법들의 소개 및 설명

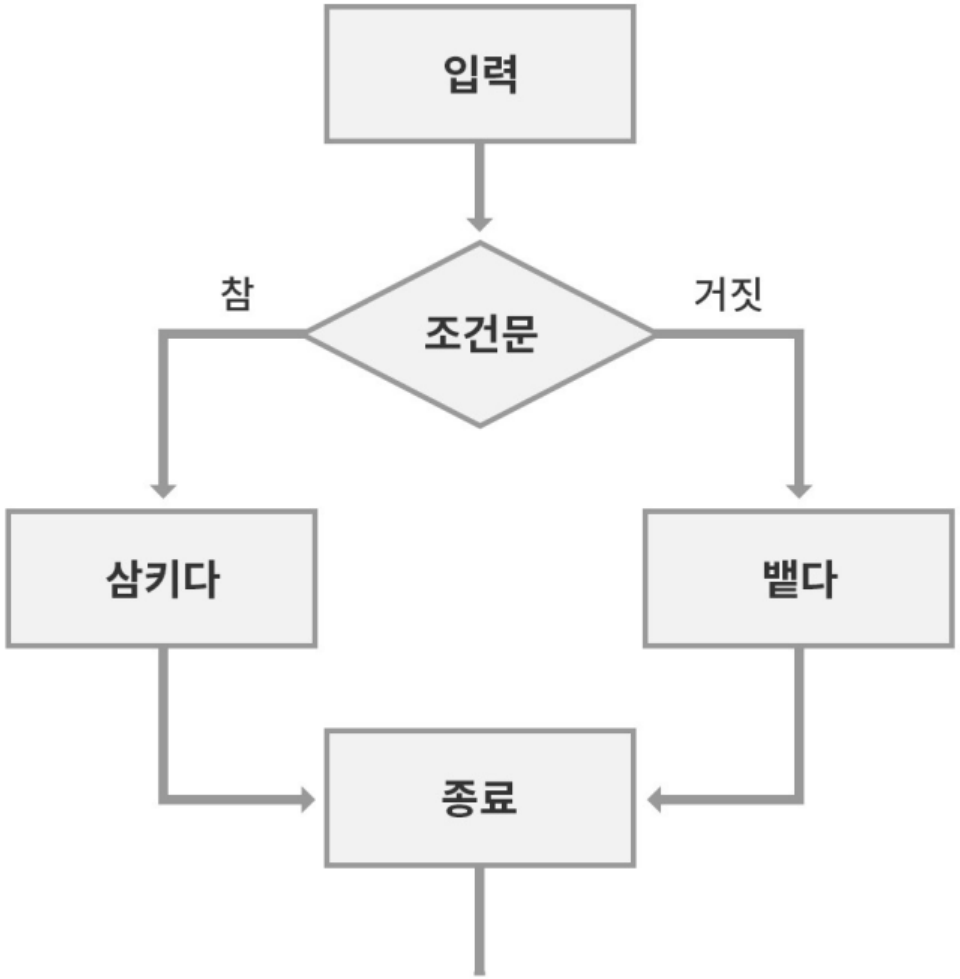
CONTENTS
INDEX

조건문

어떤 조건이 주어질 때 특정 동작을 수행하도록 하는,
즉 조건에 따라 실행을 결정할 때 사용하는 문법입니다.

01

01 조건문 (1/6)




```
if (음식이 맛있는가) {
    삼키다;
} else {
    뱉다;
}
```

❑ if 문

```c
if (number > 0)
    printf("양수입니다.\n");
```

if 문에서는 조건을 수식으로 표현하고, 그 수식을 바로 '조건식'이라고 합니다. 따라서 주어진 조건식을 계산 및 판단하여 그 결괏값에 따라 실행을 달리합니다.

❑ else 문

```
if (number > 0)
    printf("양수입니다.\n");
else
    printf("0 혹은 음수입니다.\n");
```

조건식이 거짓인 경우에도
처리를 가능하게 합니다.

❑ else if 문

```
if (number > 0)
    printf("양수입니다.\n");
else if (number == 0)
    printf("0입니다.\n");
else
    printf("음수입니다.\n");
```

다양한 조건식을 정의
하고자 할 때 사용합니다.

❑ switch 문

```c
#include <stdio.h>

int main() {
    int score = 90;
    switch(score / 10) {
        case 10:
        case 9: printf("A"); break;
        case 8: printf("B"); break;
        case 7: printf("C"); break;
        case 6: printf("D"); break;
        default: printf("F");
    } return 0;
}
```

하나의 조건식을 사용하여
발생할 수 있는 경우들을
정의해 선택을 달리합니다.

❑ goto 문

```c
int main() {
    int num = 2;
    if (num == 1) goto ONE;
    else if (num == 2) goto TWO;
    else goto EXIT;

    printf("Entered");

    ONE:
        printf("1입니다.\n");
        goto EXIT;
    TWO:
        printf("2입니다.\n");
        goto EXIT;
    EXIT:
        return 0;
}
```

```
Compiled Successfully. memory: 1632 time: 0 exit code: 0
2입니다.
```

정의된 레이블로 순서를 상관하지 않고 강제로 이동하여 코드를 실행합니다.

반복문

특정한 작업을 조건에 따라서 반복적으로
실행하는 프로그래밍 문법입니다.

02

02 반복문 (1/6)

```c
#include <stdio.h>

int main() {
    printf("1 ");
    printf("2 ");
    printf("3 ");
    printf("4 ");
    printf("5");
    return 0;
}
```

규칙적인 조건이 있을 때에
반복문을 사용하면 훨씬 간결한
코드를 작성할 수 있습니다.

02 반복문 (2/6)

❑ for 문

```c
#include <stdio.h>

int main() {
    int num = 5;
    for (int k=1; k <= num; k++)
        printf("%d ", k);
    return 0;
}
```

```
Compiled Successfully. memory: 1764 time: 0 exit code: 0
1 2 3 4 5
```

for문의 괄호 안에는
초기식[1], 조건식[2], 증감식[3]
으로 구성되어 있습니다.

❑ while 문

```c
#include <stdio.h>

int main() {
   int num = 1;
   while(num <= 5) {
      printf("%d ", num++);
   } return 0;
}
```

```
Compiled Successfully.
1 2 3 4 5
```

while 문의 괄호 안에는
조건식이 들어가야 하며,
참인 경우에만 반복을 수행합니다.

❑ do-while 문

```c
#include <stdio.h>

int main() {
    int num = 1;
    do {
        printf("%d ", num++);
    } while (num <= 5);
    return 0;
}
```

```
Compiled Successfully.
1 2 3 4 5
```

while의 괄호 안에는 조건식이 들어가며, 블록 내 코드를 한번 실행한 이후에 조건을 판단한다는 차이점이 있습니다.

❑ break 문

```c
#include <stdio.h>

int main() {
  int num = 1;
  do {
      if (num == 3) break;
      printf("%d ", num++);
  } while (num <= 5);
  return 0;
}
```

```
Compiled Successfully.
1 2
```

break 문은 곧바로 해당 반복문의 루프를 벗어나는 키워드입니다.

❑ continue 문

```c
#include <stdio.h>

int main() {
    int num = 1;
    do {
        if (num == 3) {
            num++;
            continue;
        }
        printf("%d ", num++);
    } while (num <= 5);
    return 0;
}
```

```
Compiled Successfully.
1 2 4 5
```

continue 문은 반복문의 처음 라인으로 넘어가는 키워드입니다.

함수

$$f(x)$$

하나의 특별한 목적의 작업을 수행하기 위해 독립적으로 설계된
프로그램 코드의 집합을 정의하는 문법입니다,

03

❑ 인수(argument)

```c
#include <stdio.h>

int add(int x, int y) {
    return x + y;
}

int main() {
    int result;
    result = add(5, 10);
    printf("%d", result);
}
```

함수를 호출하면서
넘겨주는 값

❑ 매개변수(parameter)

```c
#include <stdio.h>

int add(int x, int y) {
    return x + y;
}

int main() {
    int result;
    result = add(5, 10);
    printf("%d", result);
}
```

호출된 함수가 인수로부터
넘어온 값을 저장한 변수

❑ 함수원형(prototype)

```c
#include <stdio.h>

int add(int, int);

int main() {
    int result;
    result = add(5, 10);
    printf("%d", result);
}

int add(int x, int y) {
    return x + y;
}
```

컴파일러에게 함수에 대한 정보를 미리 알려주는 방법입니다.

❑ 재귀함수(recursive function)

```c
#include <stdio.h>

int fibo(int n) {
    if (n == 0) return 0;
    else if (n == 1) return 1;
    return fibo(n-1) + fibo(n-2);
}

int main() {
    int n;
    scanf("%d", &n);
    for (int k=0; k < n; k++)
        printf("%d ", fibo(k));
    return 0;
}
```

```
Compiled Successfully. memory

 0 1 1 2 3 5 8 13 21 34
```

함수 내에서 함수 스스로를
재호출하는 것입니다.

배열

동일한 자료형의 값들을 하나의 변수에 저장하여
효율적으로 관리하고자 사용하는 자료형입니다.

04

04 배열 (1/2)

❑ 1차원 배열

arr[4]

arr[0]	arr[1]	arr[2]	arr[3]

배열의 첨자는 0부터 시작합니다.

04 배열 (2/2)

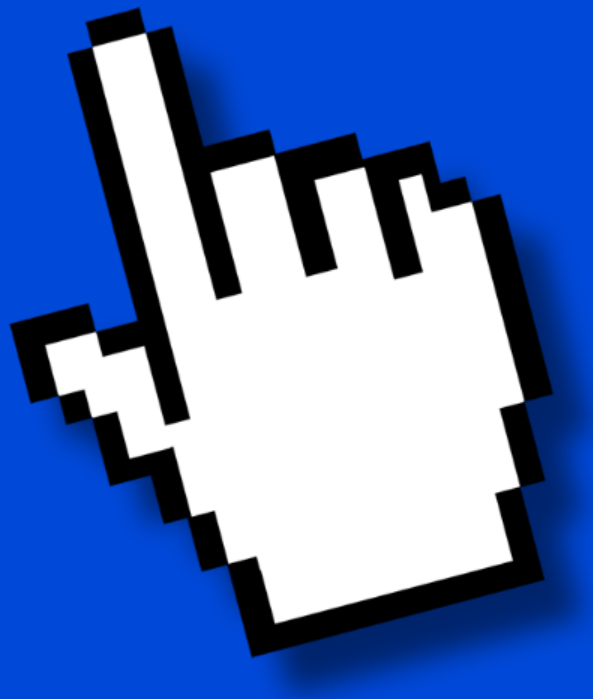
❑ 2차원 배열

arr[2][4]

arr[0][0]	arr[0][1]	arr[0][2]	arr[0][3]
arr[1][0]	arr[1][1]	arr[1][2]	arr[1][3]

2차원 배열은 '**행**' 과 '**열**'로
구성됩니다.

포인터



메모리의 주솟값을 저장하여 해당 주소에 직접적인 참조를 하고자 할 때 사용하는 문법입니다.

05

❑ 포인터

```c
#include <stdio.h>

int main() {
    int num = 10;
    int *pNum = &num;
    *pNum = *pNum + 7;
    printf("%d", num);
    return 0;
}
```

포인터 변수는 다른 변수의
주솟값을 참조합니다.

❑ 2중 포인터

```c
#include <stdio.h>

int main() {
    int num = 10;
    int *pNum = &num;
    int **dpNum = &pNum;
    *pNum = *pNum + 7;
    printf("%d\n", num);
    **dpNum = **dpNum - 5;
    printf("%d", num);
    return 0;
}
```

포인터 변수의 주솟값을
참조하는 포인터 변수입니다.

hello world
abc
programming language

문자와 문자열

글자와 문장을 변수에 저장하고자
할 때 사용하는 자료형입니다.

06

06 문자와 문자열 (1/2)

❑ 문자

```c
#include <stdio.h>

int main() {
    char c = 'A';
    printf("%c", c);
}
```

```
Compiled Successfully.
A
```

문자형은 1byte 크기의 데이터를 저장할 때 사용하는 자료형입니다.

06 문자와 문자열 (2/2)

❑ 문자열

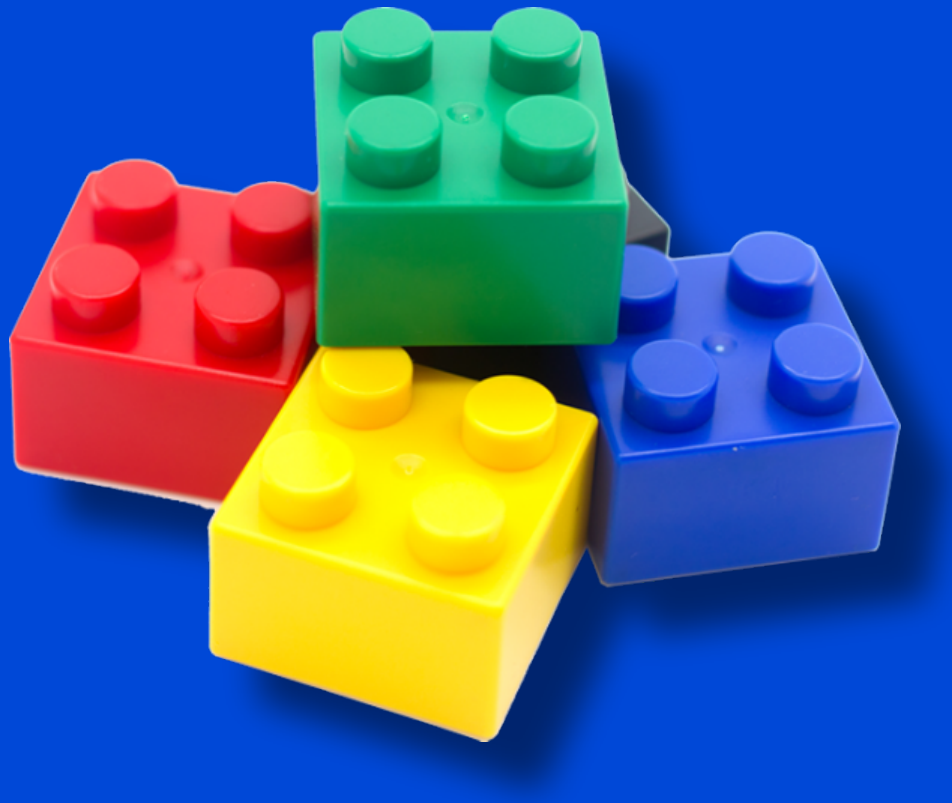
```c
#include <stdio.h>

int main() {
    char c[] = "ABC DEF";
    printf("%s", c);
}
```

```
Compiled Successfully.
ABC DEF
```

1개 이상의 문자들을 하나의 변수에 저장하고자 할 때 사용하는 문자의 배열입니다.

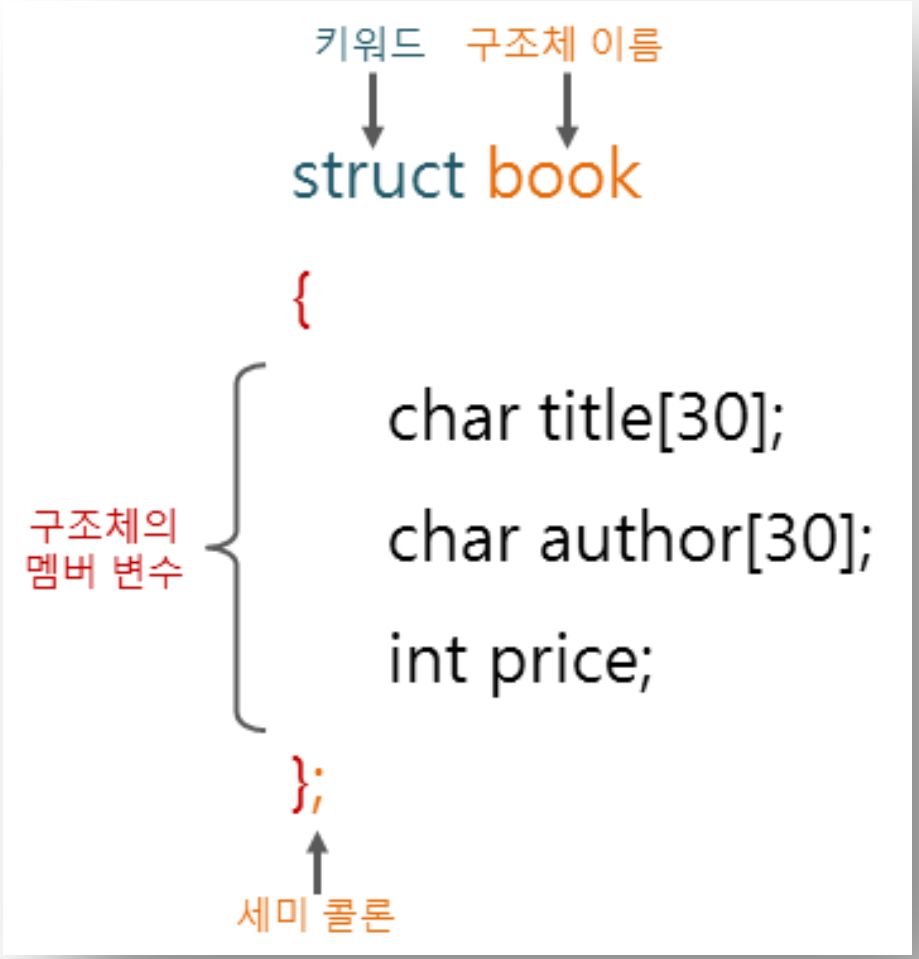
구조체

C언어에서 사용되는 기본 타입을 가지고 새롭게 정의할 수 있는 사용자 정의 타입입니다. 구조체는 기본 타입만으로는 나타낼 수 없는 복잡한 데이터를 표현할 수 있습니다.



07

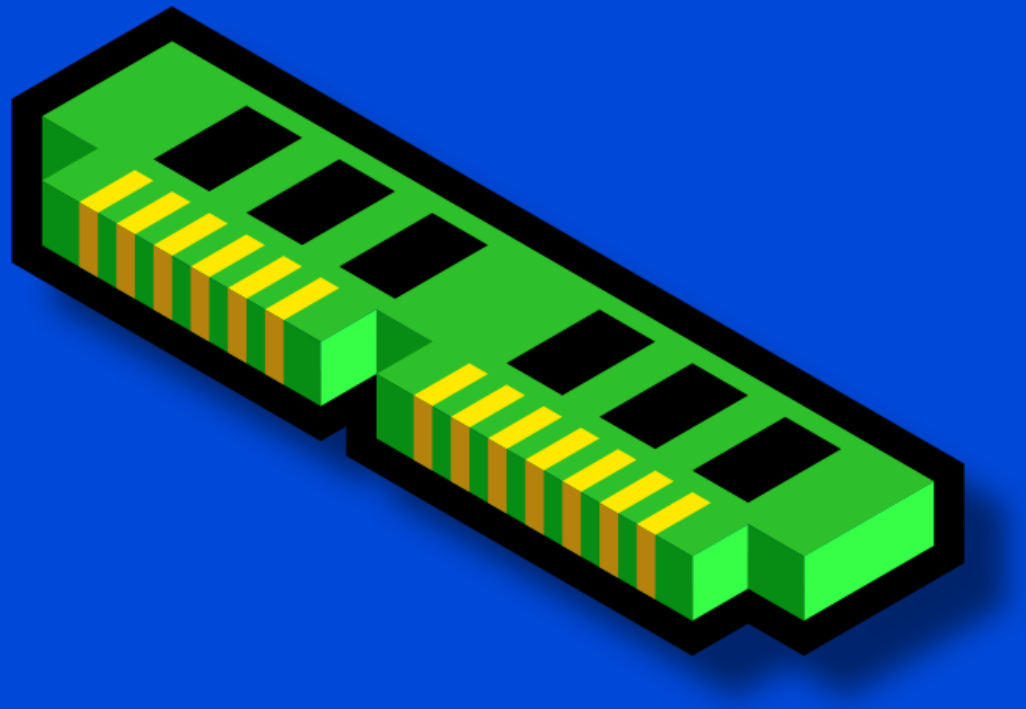
❑ 구조체



배열이 같은 타입의 변수 집합이라고 한다면, 구조체는 다양한 타입의 변수 집합을 하나의 타입으로 나타낸 것입니다.

❑ 구조체 변수 선언

```c
#include <stdio.h>

struct book {
    char title[30];
    char author[30];
    int price;
};

int main() {
    struct book myBook = { "제목", "작가", 5000 };
    printf("%s\n%s\n%d", myBook.title, myBook.author, myBook.price);
}
```



동적할당

변수를 생성하며 소모되는 메모리 공간을 보다
효율적으로 관리하기 위해 사용하는 문법입니다.

08

08 동적할당 (1/2)

❑ malloc 함수

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

int main() {
    int n, i;
    int* p;

    scanf("%d", &n);
    p = (int*)malloc(sizeof(int) * n);

    printf("%d개의 정수를 입력해주세요 : ", n);
    for (int k = 0; k < n; k++)
        scanf("%d",&p[k]);

    printf("\n입력한 정수를 역수로 출력합니다 : ");

    for (int k = n-1; k >= 0 ; k--)
        printf("%d ", p[k]);

    free(p);
    return 0;
}
```

포인터 변수와 동적할당을 함께 사용함으로써 런타임 중에 동적으로 메모리 공간을 생성할 수 있습니다.

❑ 메모리 누수

동적으로 메모리를 할당하면 힙 메모리에 공간이 생성되는데, 이는 프로그램이 종료되기 전까지 존재하여 **메모리의 낭비**를 초래해 **성능 부하**를 일으킬 수 있습니다.

QUIZ

퀴즈

간단한 문제를 통하여 이번 튜터링 시간에
익힌 내용을 실습을 통해 확인하는 시간입니다.

09

❑ 퀴즈

자연수 n을 입력으로 받아 반복문을
사용하여 아래와 같은 결과를 출력하시오.

질의응답

금일 튜터링을 진행하며 이해가 어려운 부분이 있었거나,
교과목과 관련하여 궁금한 내용을 질문하고 답변드리는
시간입니다.

10

THANKYOU

TUTORING

https://github.com/developersung13/cbnu-tutoring