
# 함수 포인터

한국기술교육대학교
온라인평생교육원

## 학습내용

- 함수 포인터 이해
- 함수 포인터 활용

## 학습목표

- 함수 포인터의 정의와 용도에 대해 설명할 수 있다.
- 함수 포인터의 기초 문법을 알고 구현할 수 있다.

# 함수 포인터 이해

## 1 기본 개념

### 1 소개

#### 1 정의

**함수 포인터란?** 함수의 주소를 저장하는 변수

#### 2 필요성

1. 프로그램 코드 간결화
2. 배열로 처리함으로써 중복 코드 제거 가능
3. 상황에 따른 함수 호출
4. 함수를 데이터 형태로 처리 →함수의 보관과 전달이 용이

# 함수 포인터 이해

## 3 형식

리턴타입 (*함수 포인터명)(매개변수리스트);

## 4 활용

### 함수 포인터의 배열

| 함수 포인터 |
| --- |
| 함수 포인터 |
| 함수 포인터 |
| 함수 포인터 |

### 구조체에 함수 포인터를 멤버로 넣음

| int age |
| --- |
| char name[20] |
| 함수 포인터 |

```
함수 포인터
반환값 자료형 함수(매개변수)
{
    return 함수 포인터;
}
```

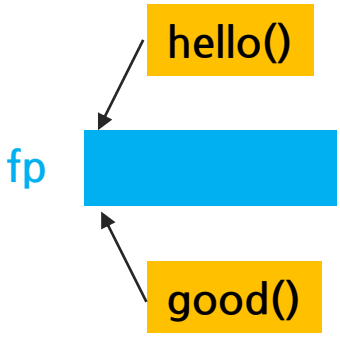
# 함수 포인터 이해

## 2 함수 포인터 형식

### 1 리턴과 매개변수가 없는 함수에 대한 함수 포인터

```
#include <stdio.h>

void hello()
{
    printf("Hello, world!₩n");
}

void good()
{
    printf("Good morning!₩n");
}
```

```
int main()
{
    void (*fp)();

    fp = hello;
    fp();

    fp = good;
    fp();

    return 0;
}
```



# 함수 포인터 이해

## 2 함수 포인터 형식

### 2 리턴과 매개변수가 있는 함수에 대한 함수 포인터

```
#include <stdio.h>

int add(int a, int b)
{
    return a + b;
}

int sub(int a, int b)
{
    return a - b;
}
```

```
int main()
{
    int (*fp)(int, int);

    fp = add;
    printf("%d₩n", fp(10, 20));

    fp = sub;
    printf("%d₩n", fp(10, 20));

    return 0;
}
```

# 함수 포인터 이해

## 2 함수 포인터 형식

### 3 함수 포인터 배열

```
#include <stdio.h>

int add(int a, int b)
{
    return a + b;
}

int sub(int a, int b)
{
    return a - b;
}
```

```
int main()
{
    int funcNumber;
    int num1, num2;
    int (*fp)(int, int) = NULL;

    printf("함수 번호와 계산할 값을 입력하세요: ");
    scanf("%d %d %d", &funcNumber, &num1, &num2);

    switch (funcNumber)    {
    case 1:
        fp = add;
        break;
    case 2:
        fp = sub;
        break;                    }

    printf("%d₩n", fp(num1, num2));

    return 0;   }
```

# 동적 메모리 이해

## 2 함수 포인터 형식

### 3 함수 포인터 배열

```
#include <stdio.h>

int add(int a, int b)
{
    return a + b;
}

int sub(int a, int b)
{
    return a - b;
}
```

```
int main()
{
    int funcNumber;
    int num1, num2;
    int (*fp[2])(int, int);

    fp[0] = add;
    fp[1] = sub;

    printf("함수 번호와 계산할 값을 입력하세요: ");
    scanf("%d %d %d", &funcNumber, &num1,
&num2);

    printf("%d₩n", fp[funcNumber](num1, num2));

    return 0;
}
```

int (*fp[2])(int, int)= { add, sub };

# 함수 포인터 이해

## 2 함수 포인터 형식

### 3 함수 포인터 배열

```
int (*fp[4])(int, int)= { add, sub, mul, div };

for (int i = 0; i < sizeof(fp) / sizeof(fp[0]); i++)
{
    printf("%d₩n", fp[i](num1, num2));
}
```

함수 호출을 데이터처럼 가능

### 4 구조체 멤버

```
#include <stdio.h>

struct Calc {
    int (*fp)(int, int);
};

int add(int a, int b)
{
    return a + b;
}
```

구조체가 함수를 포함하는 형태로 구현 가능

```
int main()
{
    struct Calc c;

    c.fp = add;

    printf("%d₩n", c.fp(10, 20));

    return 0;
}
```

# 함수 포인터 이해

## 2 함수 포인터 형식

### 5 함수의 매개변수

```
#include <stdio.h>

int add(int a, int b)
{
    return a + b;
}
```

```
void calc(int (*fp)(int, int))
{
    printf("%d₩n", fp(10, 20));
}

int main()
{
    calc(add);
    return 0;
}
```

# 함수 포인터 활용

## 1 qsort()

| 항목 | 내용 |
|---|---|
| 함수원형 | void qsort (void* base, size_t num, size_t size, int (*compar)(const void*,const void*)); |
| 헤더 | stdlib.h |
| 기능 | 테이블의 자료를 퀵 정렬로 내림이나 오름차순으로 정렬 |
| 매개변수 | void *base ➡ 테이블의 포인터 주소<br>size_t num ➡ 테이블에 들어 있는 실제 데이터 개수<br>size_t size ➡ 한 개 요소의 크기<br>int (*compar)(const void *, const void *)<br>➡ 두 요소를 비교하기 위한 함수 포인터 |
| 반환값 | void |
| 구현 | 비교함수는 직접 구현해야 함<br>(배열의 자료형과 비교방식이 다르기 때문) |

# 함수 포인터 활용

## 1 qsort()

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

int compare( const void *cmp1,
const void *cmp2)
{
   return strcmp( (char *)cmp1, (char
*)cmp2);
}

#define  SIZE_TABLE     10
#define  SIZE_ITEM      100
```

```
int main( void)
{
   char  table[SIZE_TABLE][SIZE_ITEM]  = {
      "good", "hello", "hi", "morning", "computer", "GCC", "Programming"
      };
   int   ndx;

   for ( ndx  = 0; ndx < SIZE_TABLE; ndx++)
      printf( "%s₩n", table[ndx]);

   printf( "n정렬 후n");

   qsort( table, SIZE_TABLE, SIZE_ITEM, compare);

   for ( ndx  = 0; ndx < SIZE_TABLE; ndx++)
      printf( "%s₩n", table[ndx]);

   return 0;
}
```

# 함수 포인터 활용

## 1 gsort()

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

int values[] = { 40, 10, 100, 90, 20, 25 };

int compare (const void * a, const void * b)
{
  return ( *(int*)a - *(int*)b );
}

int main ()
{
  int n;
  qsort (values, 6, sizeof(int), compare);
  for (n=0; n<6; n++)
     printf ("%d ",values[n]);
  return 0;
}
```

**오름차순**

- a < b일 때는 <0을 반환
- a > b일 때는 >0을 반환
- a == b일 때는 0을 반환

**내림차순**

- a < b일 때는 >0을 반환
- a > b일 때는 <0을 반환
- a == b일 때는 0을 반환

## 학습정리

### 1. 함수 포인터 이해

- 함수 포인터는 함수를 저장하는 포인터를 의미함
- 함수 포인터를 활용하면 함수를 자유롭게 주고받거나 함수 호출을 자동화할 수 있음
- 리턴타입 (*함수 포인터명)(매개변수리스트);

### 2. 함수 포인터 활용

- qsort는 stdlib.h를 include 해야 함
- qsort는 테이블의 자료를 퀵정렬하는 함수임
- 비교함수는 처리할 자료형에 맞게 직접 구현해야 함