

# 윤성우의 열혈 C 프로그래밍



윤성우 저 열혈강의 C 프로그래밍 개정판

Chapter 19. 함수 포인터와 void 포인터

# 윤성우의 열혈 C 프로그래밍



Chapter 19-1. 함수 포인터와 void 포인터

윤성우 저 열혈강의 C 프로그래밍 개정판

# 함수 포인터의 이해

## 1. 함수 포인터

1. **함수의 이름**은 함수가 저장된 메모리 공간을 가리키는 포인터이다(함수 포인터).
2. 함수의 이름이 의미하는 주소 값은 **함수 포인터 변수**를 선언해서 저장할 수 있다.
3. 함수 포인터 변수를 선언하려면 함수 포인터의 형(type)을 알아야 한다.

## 2. 함수 포인터의 형(type)

1. 함수 포인터의 형 정보에는 **반환형**과 **매개변수 선언**에 대한 정보를 담기로 약속
2. 즉, 함수의 반환형과 매개변수 선언이 동일한 두 함수의 함수 포인터 형은 일치한다.

## 3. 함수 포인터 형 결정

`int SimpleFunc(int num)`    반환형 **int**, 매개변수 **int형 1개**

`double ComplexFunc(double num1, double num2)`    반환형 **double**, 매개변수 **double형 2개**

# 적절한 함수 포인터 변수의 선언

```
int (*fptr) (int)
```

fptr은 포인터!

```
int (*fptr) (int)
```

반환형이 int인 함수 포인터!

```
int (*fptr) (int)
```

매개변수 선언이 int 하나인 함수 포인터!

함수 포인터 변수를 선언하는 방법

```
int SoSimple(int num1, int num2) { . . . . }
```

```
int (*fptr) (int, int);
```

*SoSimple 함수이름과 동일한 형의 변수 선언*

```
fptr=SoSimple;
```

*상수의 값을 변수에 저장*

```
fptr(3, 4);
```

*// SoSimple(3, 4)와 동일한 결과를 보임*

*함수 포인터 변수에 저장된 값을 통해서도 함수호출 가능!*

# 함수 포인터 변수 관련 예제

```
void SimpleAdder(int n1, int n2)
{
    printf("%d + %d = %d \n", n1, n2, n1+n2);
}

void ShowString(char * str)
{
    printf("%s \n", str);
}

int main(void)
{
    char * str="Function Pointer";
    int num1=10, num2=20;

    void (*fptr1)(int, int) = SimpleAdder;
    void (*fptr2)(char *) = ShowString;

    /* 함수 포인터 변수에 의한 호출 */
    fptr1(num1, num2);
    fptr2(str);
    return 0;
}
```

10 + 20 = 30  
Function Pointer

실행결과

교재에 있는 UsefulFunctionPointer.c를 통해서 함수 포인터 변수가 매개변수로 선언이 됨을 확인하기 바랍니다.

# 형(Type)이 존재하지 않는 void 포인터

```
void * ptr;
```

어떠한 주소 값도 저장이 가능한 void형 포인터

형 정보가 존재하지 않는 포인터 변수이기에 어떠한 주소 값도 저장이 가능하다.

형 정보가 존재하지 않기 때문에 메모리 접근을 위한 \* 연산은 불가능하다.

```
void SoSimpleFunc(void)
{
    printf("I'm so simple");
}

int main(void)
{
    int num=20;
    void * ptr;

    ptr=&num;    // 변수 num의 주소 값 저장
    printf("%p \n", ptr);

    ptr=SoSimpleFunc;    // 함수 SoSimpleFunc의 주소 값 저장
    printf("%p \n", ptr);
    return 0;
}
```

```
int main(void)
{
    int num=20;
    void * ptr=&num;
    *ptr=20;    // 컴파일 에러!
    ptr++;    // 컴파일 에러!
    . . .
}
```

형 정보가 존재하지 않으므로!!

001AF974

00F61109

실행결과

# 윤성우의 열혈 C 프로그래밍



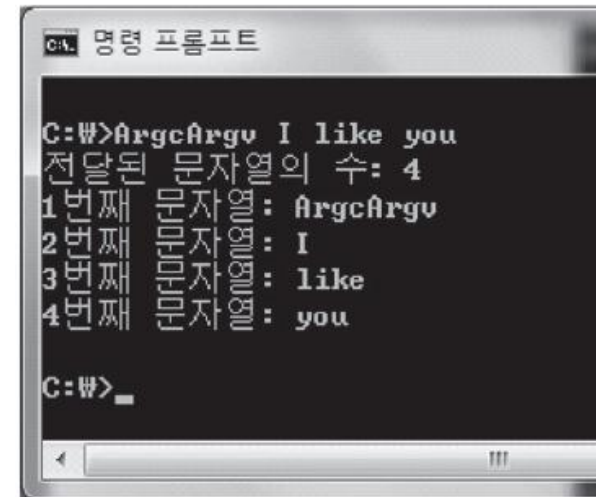
Chapter 19-2. main 함수로의 인자 전달

윤성우 저 열혈강의 C 프로그래밍 개정판

## main 함수를 통한 인자의 전달

```
int main(int argc, char *argv[])
{
    int i=0;
    printf("전달된 문자열의 수: %d \n", argc);

    for(i=0; i<argc; i++)
        printf("%d번째 문자열: %s \n", i+1, argv[i]);
    return 0;
}
```



```
C:\>ArgcArgv I like you
전달된 문자열의 수: 4
1번째 문자열: ArgcArgv
2번째 문자열: I
3번째 문자열: like
4번째 문자열: you
C:\>_
```


인자를 전달하는 방식



# char \* argv[]

```
void SimpleFunc(TYPE * arr) { . . . . }
void SimpleFunc(TYPE arr[]) { . . . . }
```

매개 변수 선언에서는 예외적으로 **\*arr**을 **arr[]**으로 대신할 수 있다!  
앞서 두 차례 확인한 내용!

 그대로 적용한다.

```
void SimpleFunc(char **arr) { . . . . }
void SimpleFunc(char * arr[]) { . . . . }
```

즉, char \* arr[]는 char형 이중 포인터이다.

## char \* argv[] 관련 예제

```
void ShowAllString(int argc, char * argv[])
{
    int i;
    for(i=0; i<argc; i++)
        printf("%s \n", argv[i]);
}

int main(void)
{
    char * str[3]={
        "C Programming",
        "C++ Programming",
        "JAVA Programming"
    };
    ShowAllString(3, str);
    return 0;
}
```

문자열의 주소 값을 모은 배열이므로 char형 포인터 배열을 선언!  
str의 포인터 형은 char\*\*

```
C Programming
C++ Programming
JAVA Programming
```

실행결과



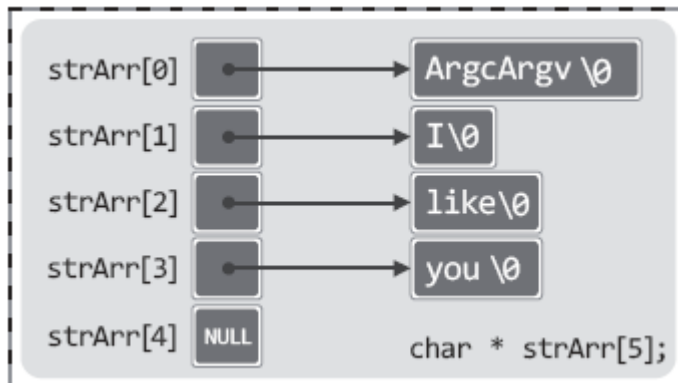
# 인자의 형성과정

c:\>ArgvArgv I like you

문자열의 구분

문자열 1	"ArgvArgv"
문자열 2	"I"
문자열 3	"like"
문자열 4	"you"

문자열의 구성



```

int main(int argc, char *argv[])
{
    int i=0;
    printf("전달된 문자열의 수: %d \n", argc);

    while(argv[i]!=NULL)
    {
        printf("%d번째 문자열: %s \n", i+1, argv[i]);
        i++;
    }
    return 0;
}
    
```

C:\> ArgvEndNULL "I love you"

전달된 문자열의 수: 2

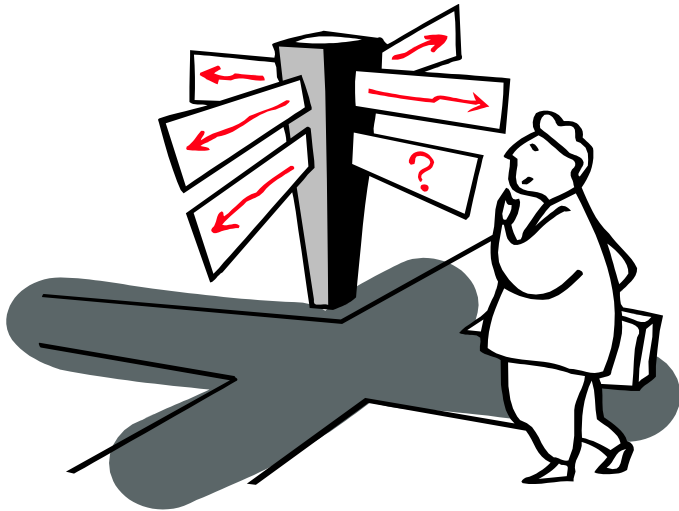
1번째 문자열: ArgvEndNULL

2번째 문자열: I love you

실행결과

문자열 기반 함수의 호출

main(4, strArr);



Chapter 19가 끝났습니다. 질문 있으신지요?