7 포인터

학습목표
- 포인터의 개념을 이해하고 포인터 사용 방법 및 유의할 점에 대해 설명할 수 있다.
- 포인터와 배열의 관계를 이해하고 포인터 연산을 적용한 프로그램을 작성할 수 있다.

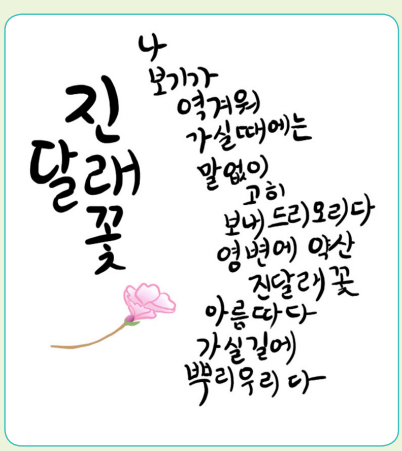
생각 펼치기

보물이나 범인을 찾는 영화를 보면, 문제를 해결할 한 단서를 찾으면 그 단서가 다음 단서로 이어지고, 그 단서를 찾으면 또 다음 단서로 이어진다. 영화에서 이와 같이 실제 보물이나 범인이 있는 위치를 가리키는 단서가 있는 것처럼 프로그래밍에서도 실제 값이 있는 위치(주소)를 가리키는 변수가 필요할 때가 있다.

핵심 질문

C 언어에서 포인터를 사용하는 방법과 포인터 연산을 적용한 프로그램은 어떻게 작성할까?

미션 이 단원을 학습하면서 해결해 보자.



김소월의 '진달래 꽃'이라는 시와 같이 긴 문자열을 저장하는 프로그램을 작성해 보자.

나 보기가 역겨워
가실 때에는
말없이 고이 보내 드리오리다.

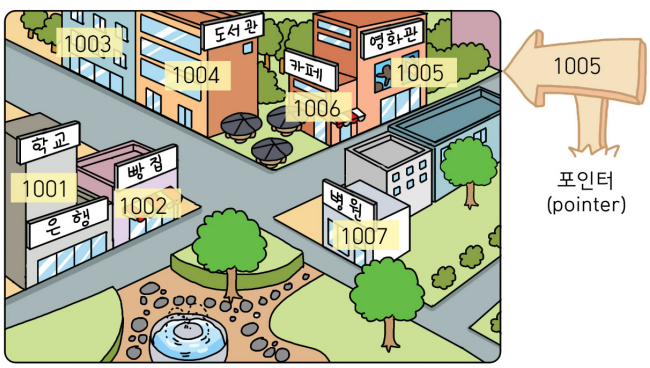
가시는 걸음 걸음
놓인 그 꽃을
사뿐히 즈려밟고 가시옵소서.

영변에 약산 ㅡ
진달래꽃
아름 따다 가실 길에 뿌리오리다.

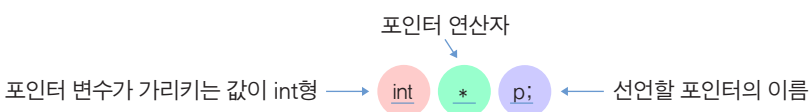
나 보기가 역겨워 ㅡ
가실 때에는
죽어도 아니 눈물 흘리오리다.

1 포인터 대해 알아보자

건물들의 주소를 보고 건물을 찾아갈 수 있듯이, 컴퓨터 메모리도 주소를 가지고 있어 그 주소로 안에 있는 데이터를 찾아간다. 이와 같이 메모리의 주소를 저장하고 있는 변수를 **포인터**라고 한다. 즉, 포인터가 저장하고 있는 것은 실제 값이 저장되어 있는 메모리의 주소이다. 아래 그림을 보면, 영화관, 학교, 빵집 등은 주소를 가지고 있지만, 포인터라는 구조물은 다른 건물의 주소만 가지고 있는 것을 알 수 있다.



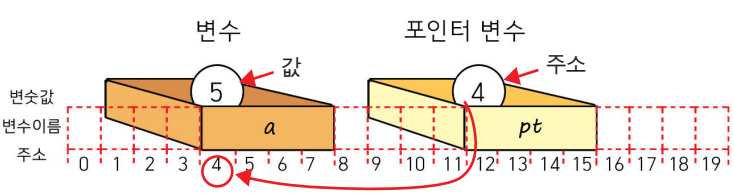
포인터를 사용하기 위해서는 포인터를 선언해야 한다. 포인터를 선언할 때에는 자료형을 먼저 쓰고, 포인터 연산자(*)를 붙인 후 이름을 쓴다. 여기서 *은 곱셈이 아니다.



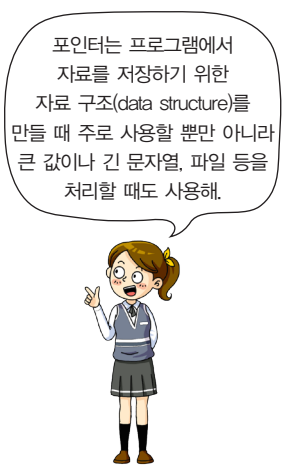
[그림 Ⅱ-8] 포인터 선언

예를 들어, 변수 a에 5를 넣어 초기화 한 후 포인터 변수 pt에 변수 a의 주소를 알려 주면 pt는 5가 있는 위치를 가리킨다.

```
int a=5;      ──── 정수형 변수 a 선언
int *pt;      ──── 포인터 변수 pt선언
pt=&a;        ──── 포인터 변수 pt에 변수 a의 주소 대입
```



[그림 Ⅱ-9] 포인터의 메모리 구조



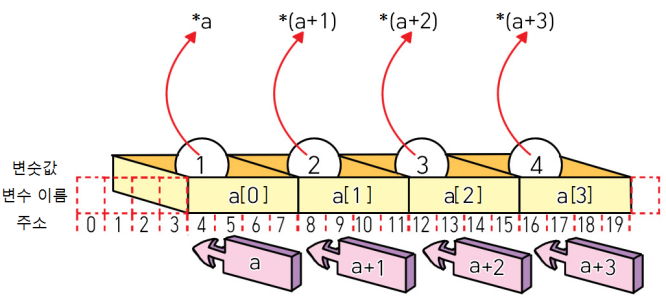
Q & A

Q 수 GByte가 넘는 동영상 파일을 변수에 저장하려고 한다. 어떻게 하면 그 큰 파일을 변수에 저장할 수 있을까?

A 수 GByte에 이르는 동영상 데이터를 일반 변수에 저장하려면 충분한 용량의 주기억 장치가 필요하지만 그렇지 못한 경우가 많다. 포인터 변수는 자료 전체를 담고 있지 않고 자료가 저장된 위치, 즉 주소만 가지고 있으므로 큰 자료도 연결하는 것이 가능하다. 즉, 주소를 이용하여 필요한 용량만 주기억 장치로 가져올 수 있는 것이다. 포인터가 없는 다른 언어도 내부적으로는 파일 등을 처리할 때 포인터와 유사한 방식을 사용한다.

2 포인터와 배열의 관계를 알아보자

배열과 포인터는 매우 밀접한 관계가 있는데 이유는 배열도 일종의 포인터이기 때문이다. 배열명은 배열의 첫 번째 요소의 주소에 해당하는 값으로, 이를 **포인터 상수**라고 한다. 따라서 배열 이름은 첫 번째 요소를 가리키는 포인터처럼 사용할 수 있다. a+1은 포인터의 덧셈 연산이다. 따라서 a+1은 a[1]을 가리키며, *(a+1)은 바로 a[1]의 내용과 같다.



[그림 Ⅱ-10] 배열과 포인터

그러나 배열의 이름이 포인터이기는 하지만 배열의 이름에 다른 변수의 주소를 대입할 수는 없다. 왜냐하면 배열의 이름은 포인터 상수로 그 값이 변경될 수는 없다.

프로그램

```
01  #include <stdio.h>
02  int main(void)
03  {
04      int a[ ] = { 1, 2, 3, 4 };
05      printf("*a = %u \n", *a);
06      printf("*(a+1) = %u \n", *(a+1));
07      printf("*(a+2) = %u \n", *(a+2));
08      return 0;
09  }
```

포인터 a가 가리키고 있는 변숫값

실행 결과

```
*a = 1
*(a+1) = 2
*(a+2) = 3
```

예제 숫자 5개가 들어 있는 배열을 포인터 변수로 받고, 포인터 값 중에 짝수의 합을 구하는 프로그램을 사용자 정의 함수를 이용하여 작성해 보자.

알고리즘 설계

① 사용자 정의 함수의 매개 변수로 포인터를 받는다.

② 포인터 연산을 이용하여 짝수를 찾는다.

③ 찾은 짝수 값을 모두 더하여 반환한다.

프로그램

```
01  #include <stdio.h>
02  int total = 0;
03  int sumpt(int *pt) {
04      int a;
05      for(a=0; a<5; a = a+1)
06          if (*(pt+a) % 2 == 0) total = total + *(pt+a);
07      return total;
08  }
09  int main(void)
10  {
11      int numpt[ ] = {89,74,36,45,98};
12      printf("%d", sumpt(numpt));
13      return 0;
14  }
```

전역 변수의 선언

배열을 포인터로 받는 사용자 정의 함수

포인터의 주소를 1씩 증가 반복

실행 결과

```
208
```

3 포인터를 함수의 매개 변수로 사용하자

함수에 매개 변수로 값을 전달할 때 값 자체를 전달하는 방법과 값이 저장되어 있는 주소를 전달하는 방식이 있다. 함수의 매개 변수로 포인터를 사용하면 값이 저장되어 있는 주소를 전달할 수 있다.

다음은 로봇이 미로를 빠져 나오는 프로그램이다. 이때 값 자체를 전달하는 robotMove( ) 함수에서는 로봇이 현재 위치에서 n칸만큼 직진하는 프로그램을 작성해도 위치 이동이 일어나지 않는다.

프로그램

```
01  #include <stdio.h>
02  void robotMove(int pxp, int pyp, int mx, int my);
03  void robotMovePt(int *pxp, int *pyp, int mx, int my);
04  int main(void)
05  {
06      int x = 1;
07      int y = 1;
08      robotMove(x, y, 1, 2);
09      printf("이동한 위치 (%d, %d) \n",  x, y);
10      robotMovePt(&x, &y, 1, 2);
11      printf("이동한 위치 (%d, %d) \n",  x, y);
12      return 0;
13  }
14  void robotMove(int pxp, int pyp, int mx, int my)
15  {
16      pxp = pxp + mx;
17      pyp = pyp + my;
18  }
19  void robotMovePt(int *pxp, int *pyp, int mx, int my)
20  {
21      *pxp = *pxp + mx;
22      *pyp = *pyp + my;
23  }
```

로봇의 현재 위치 x, y

위치 이동 함수 호출

포인터를 매개 변수로 활용한 함수 호출

실행 결과

```
이동한 위치 (1, 1)
이동한 위치 (2, 3)
```

앞의 프로그램에서 robotMove 함수를 호출하면 변수의 값만 전달되었으므로 원래 변수의 값은 변하지 않는다. 왜냐하면 robotMove 함수의 매개 변수로 넣은 변수 x, y값은 값이 함수로 전달되는 것이므로, main 함수의 x, y 변수와 robotMove 함수의 매개 변수는 다른 값이기 때문이다. 이런 함수 호출 방식을 **값에 의한 호출**(call by Value)라고 한다. 만약 원래 변수의 값을 바꾸려면 변수의 주소를 참조하여 변수의 값을 바꾸어야 한다. 이러한 방법을 **참조에 의한 호출**(Call by Reference)이라고 한다.

예제 **변수 2개의 값을 교환하는 함수 swap( )를 작성해 보자.**

알고리즘 설계

① 포인터를 매개 변수로 하여 값을 받는다.

② 두 값을 바꾸어 반환한다.

프로그램

```
01  #include <stdio.h>
02  void swap(int *a, int *b);
03  int main(void)
04  {
05      int a, b;
06      a = 5;
07      b = 7;
08      printf("함수 호출 전 : a = %d, b = %d \n", a, b);
09      swap(&a, &b);
10      printf("함수 호출 후 : a = %d, b = %d \n", a, b);
11      return 0;
12  }
13  void swap (int *a, int *b)
14  {
15      int temp;
16      temp = *a;
17      *a = *b;
18      *b = temp;
19  }
```

a와 b의 주소 전달

호출 후 a, b 값

실행 결과

```
함수 호출 전 a=5, b=7
함수 호출 후 a=7, b=5
```

4 포인터로 문자열을 처리하는 방법을 알아보자

C 언어에서 다소 긴 문자열을 지정하는 방법은 배열을 이용하는 방법과 포인터를 이용하는 방법이 있다. 포인터는 배열과 달리 초기화한 후에 '=' 연산자를 이용해 전체 값을 다시 저장할 수 있다.

포인터 배열

배열의 요소가 포인터인 배열이다. 같은 형의 포인터가 여러 개 필요할 때 사용한다. 예를 들어, January부터 December까지 1년 12달의 이름을 한 변수에 저장할 때 유용하게 사용할 수 있다.

```
char *p = "HelloWorld";   // 포인터를 정의하고 문자열의 주소로 포인터 초기화
p = "Goodbye";            // "Goodbye"가 저장된 주소로 포인터의 값 변경
```

포인터를 이용해서 여러 개의 문자열을 저장하려면 포인터 배열을 이용하면 된다. 이 방법은 한 번 입력 후 바뀌지 않는 문자열을 저장할 때 사용한다.

실행 결과

```
여러분이 배워야할
언어는 C JAVA
PYTHON
```

프로그램

```
01  #include <stdio.h>
02  int main(void)
03  {
04      char *lang[3] = { "C", "JAVA", "PYTHON" };   // 포인터 배열 사용
05      int a;
06      printf("여러분이 배워야 할 언어는");
07      for (a = 0; a < 3; a++) printf(" %s ", lang[a]);
08      return 0;
09  }
```

미션 해결하기

김소월의 '진달래 꽃'이라는 시와 같이 긴 문자열을 저장하는 프로그램을 작성해 보자.

프로그램

```
01  #include <stdio.h>
02  int main(void)
03  {
04      char *p = "진달래꽃     김소월\n\n나 보기가 역겨워\n가실 때에는\n말없이 고이 보내 드리오리다.\n\n영변에 약산\n
                  진달래꽃\n아름 따다 가실 길에 뿌리오리다.\n\n가시는 걸음걸음\n놓인 그 꽃을\n사뿐히 즈려 밟고 가시옵소
                  서.\n\n나 보기가 역겨워\n가실 때에는\n죽어도 아니 눈물 흘리오리다.";
05      printf("%s \n", p);
06      return 0;
07  }
```

5 문자열 입출력 함수를 사용해 보자

이제까지 문자열을 입력받을 때에는 scanf( ) 함수를 사용하고 출력할 때에는 printf( ) 함수를 사용하였다. 그러나 scanf( ) 함수는 공백이 있는 문자열을 입력받을 수 없다. 이때 사용하는 라이브러리 함수가 gets( )이다.

[표 Ⅱ-12] 문자열 입출력 함수

입출력 함수	설명
char *gets(char *s)	한 줄의 문자열을 읽어서 문자 배열 s[ ]에 저장한다.
int puts(const char *s)	배열 s[ ]에 저장되어 있는 한 줄의 문자열을 출력한다.

포인터와 문자형 배열

문자형 포인터는 문자형 배열과 동일하게 취급할 수 있다. 즉 다음 두 문장은 동일하다.

```
char *s="ABC";
char s[ ]="ABC";
```

예제 gets( )와 puts( ) 함수를 이용하여, 주소를 입력받아 출력하는 프로그램을 작성해 보자.

프로그램

```
01  #include <stdio.h>
02  int main(void)
03  {
04      char address[100];
05      char *addpt;
06      printf("주소 입력: ");
07      gets(address);
08      addpt = address;
09      printf("입력한 주소는: ");
10      puts(addpt);
11      return 0;
12  }
```

실행 결과 예

```
주소 입력: 서울특별시 마포구 삼개로
입력한 주소는: 서울특별시 마포구 삼개로
```

예제 입력된 문자열에서 '대문자, 소문자, 숫자가 각각 1개 이상 없거나 6자 미만인 경우 암호를 다시 만들라'는 메시지를 출력하는 프로그램을 작성해 보자.

알고리즘 설계

① 6자리 이상 20자리 이하의 문자열을 입력받는다.

② 6자리가 안 되거나, 대문자가 없거나 소문자 및 숫자가 없으면 암호를 다시 만들라고 출력한다.

③ 조건을 만족하면 해당 암호를 출력한다.

프로그램

strlen( ) 함수

strlen 함수는 문자열의 길이를 반환하는 함수이다. 함수의 매개 변수에 문자열을 입력하면 그 문자열의 길이를 반환한다.
예 printf("%d", strlen("abcde"));
문장을 실행하면 5를 출력한다.

```
01  #include <stdio.h>
02  #include <string.h>
03  int main(void)
04  {
05    int i;
06    int isUpper, isLower, isDigit;
07    char mypasswd[20];
08    while (true) {
09      printf("사용할 암호를 넣으세요: ");
10      scanf("%s", mypasswd);
11      isUpper = isLower = isDigit = 0;
12      for (i = 0; i < strlen(mypasswd); i++) {
13          if (mypasswd[i] >= '0' && mypasswd[i] <= '9')
14              isDigit++;
15          else if (mypasswd[i] >= 'A' && mypasswd[i] <= 'Z')
16              isUpper++;
17          else if (mypasswd[i] >= 'a' && mypasswd[i] <= 'z')
18              isLower++;
19      }
20      if ((i >= 6) && (isDigit >= 1) && (isUpper >= 1) &&
          (isLower >= 1)) {
21        printf("암호로 사용 가능합니다. \n");
22        break;
23      }
24      else
25      printf("6문자 이상, 숫자, 대문자, 소문자가 각 1개 이상 있어야 합니다.\n");
26    }
27    return 0;
28  }
```

실행 결과

```
사용할 암호를 넣으세요: Qwerty1!
암호로 사용 가능합니다.
```

스스로 해결하기

포인터 배열을 이용하여, 24절기 중 봄에 해당하는 6절기(입춘, 우수, 경칩, 춘분, 청명, 곡우)를 저장하여 출력하는 프로그램을 작성해 보자.

알고리즘 설계

① 포인터 배열에 6절기를 입력하여 초기화한다.
② 반복문을 이용하여 6절기를 출력한다.

프로그램

```
01  #include <stdio.h>
02  int main( )
03  {
04     int a;
05     char *spring_term[6] = {"입춘", "우수", "경칩", "춘분", "청명", "곡우"};
06     for (a = 0; a < 6; a++) {
07          printf("%s ", spring_term[a]);
08     }
09     return 0;
10  }
```

24 절기

- 봄: 입춘, 우수, 경칩, 춘분, 청명, 곡우
- 여름: 입하, 소만, 망종, 하지, 소서, 대서
- 가을: 입추, 처서, 백로, 추분, 한로, 상강
- 겨울: 입동, 소설, 대설, 동지, 소한, 대한

응용하기

scanf( ) 함수를 이용하여, 절기 이름을 입력받아 그것이 '봄'에 해당하는 절기이면 "봄 절기 입니다."를 출력하는 프로그램을 작성해 보자.

알고리즘 설계

① 절기 이름을 입력받은 변수는 배열 변수 tname[10]이라고 한다.
② 문자열 비교는 strcmp( ) 함수를 이용한다.

프로그램

```
01  #include <stdio.h>
02  #include <string.h>
03  int main( )
04  {
05       int a;
06       char *spring_term[6] = {"입춘", "우수", "경칩", "춘분", "청명", "곡우"};
07       char tname[10];
08       scanf("%s",tname);
09       for (a = 0; a < 6; a++) {
10          if (strcmp(spring_term[a], tname)==0) {
11             printf("%s 봄 절기 입니다. ", tname);
12             break;
13          }
14       }
15       return 0;
16  }
```

strcmp( ) 함수

int strcmp(const char *s1, const char *s2);

위와 같은 형태로 사용하고 문자열 두 개가 같으면 숫자 0을 반환한다. 그렇지 않으면 0이 아닌 숫자를 반환한다.

컴퓨팅 사고력을 키우는

수행 평가 포인터 배열을 이용한 문자열 처리

활동 목표 포인터 배열을 사용하여 자료를 입력하고 출력할 수 있다.

다음은 2차원 배열에 문자열을 저장하는 프로그램이다.

프로그램

```
01  #include <stdio.h>
02  int main(void)
03  {
04      int a;
05      char animals[3][10] = { "dog", "cat", "lion" };
06      for (a = 0; a < 3; a++)
07          printf(" %d 번째 동물은 %s 입니다\n", a+1, animals[a]);
08      return 0;
09  }
```

초급 위 프로그램을 실행시켰을 때, 출력되는 값을 예측하여 적어 보자.

중급 2차원 배열 변수 animal을 포인터 배열로 변경하여 프로그램을 작성해 보자.

스스로 평가하기

평가 항목	구분		
	그렇다	보통이다	그렇지 않다
• 배열의 구조를 이해하여 값을 저장할 수 있다.			
• 배열에서 행과 열를 바꾸어 저장하는 프로그램을 작성할 수 있다.			

내 실력 확인하기

내용을 이해했나요?

- **메모리 주소**: 컴퓨터의 기억 장소인 메모리의 값을 읽거나 쓰기 위한 식별자를 말하며, 프로그램은 이 주소를 이용하여 메모리에 있는 특정 값에 접근한다.
- **포인터 변수**: 메모리의 주소를 값으로 가지고 있는 변수이다.
- **포인터 상수**: 배열의 첫 번째 요소의 주소에 해당하는 값이다.
- **포인터 배열**: 배열과 포인터를 동시에 사용하는 변수이다.

문제로 확인할까요?

1. 포인터를 사용하여 문자열을 저장할 때, 배열을 이용한 것보다 어떤 장점이 있는지 적어 보자.

_____

2. 다음 중 정수형 포인터를 선언한 것으로 옳은 것은?

① int a[10]; ② int *a; ③ int &a; ④ int ^a; ⑤ int a;

3. 다음 프로그램을 실행했을 때 출력되는 값을 적어 보자.

```
01  #include <stdio.h>
02  int main(void)
03  {
04      int a;
05      char *pt = "I love you";
06      for (a=7;a<10;a++)
07          printf("%c", *(pt+a));
08      return 0;
09  }
```

실행 결과

평가해 볼까요?

★다음 평가 항목에 따라 자신의 성취 척도를 스스로 점검해 보자.

영역	평가 항목	척도 1	2	3	4	5
이해	포인터 기능과 형식을 설명할 수 있는가?					
적용	여러 문자 열을 저장할수 있는 포인터 배열을 선언하고 사용할 수 있는가?					