포인터 변수 생성 및 응용

Main Category	Log
Category	Algorithm
Tags	C
Status	Done
Created Time	@September 16, 2023 11:27 PM
Updated Time	@September 17, 2023 11:30 PM

포인터 변수 생성 및 응용

포인터(pointer): 메모리상의 주소를 담고 있는 변수

https://discreet-bay-25b.notion.site/3c7a969e319a44a58f7f210bf335f567?pvs=4

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/c02edec9-e488-45fe-93f1-d950ff3f5cd5/ch11_%E1%84%91%E1%85%A9%E1%84%8B%E1%85%B5%E1%86%AB%E1%84%90%E1%85%A5.pdf.pdf

포인터(pointer): 메모리상의 주소를 담고 있는 변수

```
char* pc = NULL; //문자를 가리키는 포인터 pc
char c = 'A'; //문자형 변수 c

pc = &c; //포인터 pc에 c의 주소를 저장

printf("%u %u\n",pc, &c);
//-> 127000 127000 포인터변수의 값과 c변수의 주소값이 같음

printf("%c",*pc);
//-> A  간접 참조 연산자 * : 포인터가 '가리키는 곳'의 '값'을 의미하는 연산자
```

& 연산자와 * 연산자 비교

& 연산자 : (일반 변수에 붙임) 변수의 주소를 반환한다.

* 연산자 : (포인터에 붙임) 포인터가 가리키는 곳의 값을 반환한다.

포인터 연습 11-1

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/f1c2af79-faea-49aa-b4cc-0ab2241f0dd0/%EC%B6%9C%EB%A0%A5.pdf

구현 코드

```
#include <stdio.h>

int main(int argc, char* argv[]) {
  int i, j;
  int *pi, *pj;
  pi = &i;
  pj = &j;

  double avg;
  double *pavg;

  pavg = &avg;

  *pi = 5;
  *pj = 10;

  *pavg = (double)(*pi + *pj)/2;

  printf("%d %d %f\n",*pi,*pj,*pavg);
  printf("%d %d %f\n",i,j,avg);

  return 0;
}
```

포인터 사용시 주의점

포인터가 아무것도 가리키고 있지 않는 경우에는 반드시 NULL로 초기화한다.

초기화가 안된 포인터를 사용하면 안된다.

```
#include <stdio.h>
int main(void)
{
  int* p = NULL; //포인터 생성시에 NULL로 초기화
  *p = 100; //위험한 코드
  return 0;
}
```

포인터의 타입과 변수의 타입은 일치하여야 한다.

```
#include <stdio.h>
int main(void)
{
  int i;
  double *pd;

  pd = &i; // 오류! double형 포인터에 int형 변수의 주소를 대입
  *pd = 36.5;
```

```
return 0;
}
```

포인터 연산

가능한 연산 : 덧셈, 뺄셈 연산 (포인터가 가리키는 주소를 앞 뒤로 이동)

증가 연산의 경우, 포인터가 가리키는 객체의 크기(자료형의 바이트 크기)만큼 증가된다.

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/b2fcab23-0c80-49d3-8dda-46812210c919/%EC%B6%9C%EB%A0%A5.pdf

간접 참조 연산자와 증감 연산자

괄호를 필수적으로 사용하여 의미를 명확화해야한다.

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/acb850b6-0166-44c4-92ce-e1d0980caa62/%EC%B6%9C%EB%A0%A5.pdf

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/1eaca5b8-adcd-4609-b1f0-f8b998fb3514/%EC%B6%9C%EB%A0%A5.pdf

호출자가 함수로부터 값을 전달받는 방법 (2개 이상의 결과를 반환할 때 많이 사용함)

1. return value;
2. 호출자가 함수에게 값을 전달 받을 장소의 pointer를 넘겨주어서,
   함수가 그 곳에 값을 채워서 호출자에게 넘겨 준다.

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/7ef3b659-1d59-4062-8562-bceebd23a313/%EC%B6%9C%EB%A0%A5.pdf

포인터와 배열

배열 이름은 포인터처럼 사용할 수 있다.

또한, 포인터는 배열처럼 사용할 수 있다.

예제 - 배열이름을 포인터처럼 사용

```
#include <stdio.h>
int main(void)
{
  int a[5] = { 10, 20, 30, 40, 50 };
  printf(“a = %u\n”, a);
  printf(“a + 1 = %u\n”, a + 1);
  printf(“*a = %d\n”, *a);
  printf(“*(a+1) = %d\n”, *(a+1));
  *(a+2) = 60;
  printf(“a[2] = %d\n”, a[2]);
  return 0;
}
```

함수 내에서 두 개 이상의 배열 값을 반환해야할 때

포인터 선언 필요 없이, 배열 이름으로 포인터처럼 바로 반환 가능

예제 - 포인터를 배열처럼 사용

```
#include <stdio.h>
int main(void)
{
  int a[] = { 10, 20, 30, 40, 50 };
  int *p;

  p = a;
  printf("a[0]=%d a[1]=%d a[2]=%d \n", a[0], a[1], a[2]);
  printf("p[0]=%d p[1]=%d p[2]=%d \n\n", p[0], p[1], p[2]);
  //배열은 결국 포인터로 구현된다는 것을 알 수 있다.

  p[0] = 60;
  p[1] = 70;
  p[2] = 80;

  printf("a[0]=%d a[1]=%d a[2]=%d \n", a[0], a[1], a[2]);
  printf("p[0]=%d p[1]=%d p[2]=%d \n", p[0], p[1], p[2]);
  //포인터을 통하여 배열 원소를 변경할 수 있다.

  return 0;
}
```

p[2] = 80; 이후에 p[2] == a[2] 임을 볼 수 있듯, 포인터를 통해 배열 원소를 바로 변경할 수 있다.

함수에서 배열이름을 매개변수로 받아 포인터처럼 사용 가능

```
#include <stdio.h>
void sub(int b[], int n);
int main(void)
{
  int a[3] = { 1,2,3 };

  printf("%d %d %d\n", a[0], a[1], a[2]);

  sub(a, 3);
  printf("%d %d %d\n", a[0], a[1], a[2]);

  return 0;
}
void sub(int *b, int size)
{
  *b = 4;
  *(b+1) = 5;
  *(b+2) = 6;
}
```

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/c73cf78c-8509-4e5a-95f4-c1b3c89e2028/%EC%B6%9C%EB%A0%A5.pdf

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/c69a2a3b-a277-44da-aa54-1131ecbeeed4/%EC%B6%9C%EB%A0%A5.pdf

숙제11-2 : 영상 처리

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/075d93be-48c5-455b-9132-f7f81e792029/%EC%B6%9C%EB%A0%A5.pdf

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/af6aabdb-dcaf-427e-b1ac-c8a87e2e876f/%EC%B6%9C%EB%A0%A5.pdf

```
#include <stdio.h>
#define SIZE 5

void print_image(int img[][SIZE], int xsize, int ysize);
void brighten_image(int img[][SIZE], int xsize, int ysize);
```

```
int main(void)
{
  int image[SIZE][SIZE] = {
  { 10, 120, 130, 240, 250},
  { 10, 120, 130, 240, 250},
  { 10, 120, 130, 240, 250},
  { 10, 120, 130, 240, 250},
  { 10, 120, 130, 240, 250} };

  print_image(image, SIZE , SIZE);
  brighten_image(image, SIZE , SIZE);
  print_image(image, SIZE , SIZE);
  return 0;
}

void print_image(int img[][SIZE], int xsize, int ysize){
  for(int i=0; i<ysize; i++){
    for(int j=0; j<xsize; j++){
      printf("%d ",*(*(img+i)+j));
    }
    printf("\n");
  }
}
void brighten_image(int img[][SIZE], int xsize, int ysize){
  for(int i=0; i<ysize; i++){
    for(int j=0; j<xsize; j++){
      *(*(img+i)+j) += 10;
      if(*(*(img+i)+j) > 255) *(*(img+i)+j) = 255;
    }
  }
}
```

설계 방법 요약

이차원 배열의 값에 접근하여 수정하는 것이 목표인 과제였다.

배열 표기법을 사용하여 이차원 배열에 접근하면 훨씬 쉬웠겠지만,

배열과 포인터의 교차사용에 의미를 두기 위해서 포인터 주소로만 접근하는데 중점을 뒀다.

이차원 배열의 구조는 일차원 배열 각 인덱스의 '값'에 다른 일차원 배열의 '주소'가 있는 구조임에 착안했다.

1. 바깥 일차원 배열의 주소를 매 반복마다 1씩 증가시키고, 그 주소의 값을 호출한다.
   → *(img+i) (&img[i]와 근본적으로 동일)
2. *(img+i)는 다른 일차원 배열의 주소이므로 그 배열의 주소를 매 반복마다 1씩 증가시킨다.
   → *(img+i)+j (&img[i][j]와 근본적으로 동일)
3. 안쪽 일차원배열의 그 주소의 값을 호출한다. → *(*(img+i)+j) (img[i][j]와 근본적으로 동일)
4. 최종적으로 *(*(img+i)+j) + 10 은 img[i][j] + 10 과 근본적으로 동일한 포인터식 표현방법이다.

Lab: 자율 주행 자동차

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/4bf6fa43-15ad-495d-8ffa-d911262ed2c8/%EC%B6%9C%EB%A0%A5.pdf

```
#include <stdio.h>

// 0부터 99까지의 난수(실수형태)를 발생하여 크기가 3인 배열 p에 저장한다.
void getSensorData(double * p)
{
  *p = rand()%100;
  *(p+1) = rand()%100;
  *(p+2) = rand()%100;
  return;
}

int main(void)
{
  double sensorData[3];
  getSensorData(sensorData);
  printf("왼쪽 센서와 장애물과의 거리: %lf \n", sensorData[0]);
  printf("중간 센서와 장애물과의 거리: %lf \n", sensorData[1]);
  printf("오른쪽 센서와 장애물과의 거리: %lf \n", sensorData[2]);
  return 0;
}
```

p[0], p[1] 등 배열 표기법으로 구현할 수 있었겠지만, [배열↔포인터] 연습을 위해 포인터 식으로 구현하였다.

숙제 11-3 : 포인터를 이용한 성적처리

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/c9008ba1-6909-4672-8567-cda291061e15/%EC%B6%9C%EB%A0%A5.pdf

```
#include <stdio.h>
#define SIZE 5

void calculate(double *e, double *m, double *e100, double *m100, double *av, int size);

int main(void)
{
  double eng[SIZE] = {4.1, 3.0, 2.8, 4.2, 3.5};
  double math[SIZE] = {3.1, 3.5, 3.3, 3.2, 2.7};
  double eng100[SIZE];
  double math100[SIZE];
  double avg[SIZE];

  calculate(eng,math,eng100,math100,avg,SIZE);

  for(int i=0; i<SIZE; i++){
    printf("%d번 학생 영어백분율: %.1f, 수학백분율: %.1f, 평균백분율: %.1f\n",(i+1),eng100[i],math100[i],avg[i]);
  }

  return 0;
}

void calculate(double *e, double *m, double *e100, double *m100, double *av, int size){
  for(int i=0; i<size; i++){
    e100[i] = (e[i]/4.5)*100;
    m100[i] = (m[i]/4.5)*100;
    av[i] = (e100[i]+m100[i])/2.0;
```

```
  }
  return;
}
```

아래는 위 코드 calculate 함수를 포인터 방식으로 구현한 것.

```
void calculate(double *e, double *m, double *e100, double *m100, double *av, int size){
  for(int i=0; i<size; i++){
    *(e100+i) = (*(e+i)/4.5)*100;
    *(m100+i) = (*(m+i)/4.5)*100;
    *(av+i) = (*(e100+i)+*(m100+i))/2.0;
  }
  return;
}
```

최종적으로 배운 내용

- 배열과 포인터를 자유자재로 오가며 쓸 수 있다.
- 포인터 방식을 사용하여 함수 내 여러개 값을 반환할 수 있다.
- 이차원 배열의 메모리 구조를 이해할 수 있다.
- 이차원 배열을 포인터 방식으로 활용할 수 있다.