

Work_Log_2024_07_09

C언어 7일차 과제

작성자 : 박기홍

Remind

Remind

1. 로빏에 입단한 후 저는 군사용 로봇과 구조용 로봇을 개발하여 사람들의 안전과 생명을 지키는 데 기여하는 것을 목표로 임하겠습니다.
2. 지도교수님과 선배님들을 공경하며 동기들을 존중하는 마음가짐으로 임하겠습니다.
3. 로빏에서 배운 공학적 기술과 경험은 사람들을 살리는 데 사용될 수 있으며, 악한 마음을 품게 된다면 사람들에게 피해를 줄 수 있다는 것을 명심하면서 올바른 로봇 엔지니어로서 성장할 수 있도록 윤리를 중요시하겠습니다.
4. 글로벌 로봇 산업의 성장기와 성숙기를 이끌어 나아가는 엘리트로 성장할 수 있는 능력을 만들겠습니다.

TODO List

Info

어떤 것을 할지 미리 생각해 놓는 시간입니다. 간략하게 2~5개로 적습니다.

- ✓ 옴사디언 생성하기
- ✓ 자료구조 공부하기(Do it! C언어 자료구조)
- ✓ 백준 문제 풀기
- ✓ 깃허브 ReadMe.md 꾸미기
- ✓ 과제하기

Activity

Info

오늘 작업한 내용을 작성합니다.

- 개발을 하면서 고민한 부분
- 개발을 하면서 참고한 문서
- 새로 알게 된 사실 등의 내용

1. 옴시디언 생성하기

Start Time : 13:47

End Time : 13:48

2. 자료구조 공부하기(Do it! C언어 자료구조)

Start Time : 14:03

End Time : 14:30

전체 범위 : p.337부터 ~ p.401

시작 범위 : p.337

종료 범위 : p.346

연결 리스트를 공부하기 위해 해당 범위를 우선 공부했다. 공부해도 어렵게 느껴진다. 계속해서 봐야 할 것 같다.

3. 백준 문제 풀기

Start Time : 14:30

End Time : 16:30

백준에 게시된 쉬운 문제들을 풀었다. 나는 실버 5 랭크에 도달했다. 쉬운 문제만 풀어서, 이제 난이도 있는 문제들을 풀어야 할 것 같다.

로봇 수습 단원 프로세스에 참가하기 이전에는 못 풀던 문제를, 오늘 풀 수 있었다. 로봇에서 배정 받는 과제를 수행해서 그런지 그동안 못 풀던 백준 문제가 쉽게 풀렸다. 행복하다.

4. Github ReadMe.md 꾸미기

Start Time : 16:30 / 19:00

End Time : 18:00 / 19:15

Github Profile(ReadMe.md) 파일을 수정했다.

예쁘게 디자인 했다.

마크다운 문법은 역시 재밌다.

5. 과제하기

Start Time : 21:05 / 2024.07.10. 09:30(과제3 시작)

End Time : 2024.07.10. 04:03 (과제1까지) / 2024.07.10. 19:50(과제3 임시 종료)

Today's Completion

 Info

오늘 완료한 작업을 정리합니다.

1. 옵시디언 생성하기
2. 자료구조 공부하기
3. 백준 문제 풀기
4. Github ReadMe.md 꾸미기
5. 과제하기

Today's Concept

Concept1

Concept2

Solving a HomeWork

Info

과제 풀이를 작성합니다.

과제1

과제 설명

포인터에 대해(개념, 배열, 함수 등등)에 대한 복습 및 공부한 후 해당 내용을 정리하여 제출. 형식 상관 X, 보고서, 그림, 혼자 끄적이면서 공부한내용 다 OK.

[추가 설명]

1번 과제에 포함해야 하는 사항

15쪽 **각 함수별 작동원리** 정리해서 제출(종이에 그림을 그려 스캔 후 제출, ppt로 만들어서 제출.. 등 제출 양식은 자유)

25쪽 **이중포인터가 포함된 함수의 작동원리**에 대해 서술하시오.(자유양식)

과제 1



포인터에 대해(개념, 배열, 함수 등등)에 대한 복습 및 공부한 후 해당 내용을 정리하여 제출.
형식 상관 X, 보고서, 그림, 혼자 끄적이면서 공부한내용 다 OK.

과제1-1

과제 설명

포인터에 대해(개념, 배열, 함수 등등)에 대한 복습 및 공부한 후 해당 내용을 정리하여 제출. 형식 상관 X, 보고서, 그림, 혼자 끄적이면서 공부한내용 다 OK.

포인터 (Pointer)

포인터 : 어떤 다른 값이 저장되어 있는 메모리 주소를 가지고 있는 상수/변수.

Important

1. 포인터 변수는 자신의 자료형에 맞는 주소 값을 참조할 수 있음.
2. 초기화 되지 않은 포인터 변수에 접근할 수 없음.

Why use pointers

1. 함수 내부에서 함수 외부의 값을 가져오거나 수정할 수 있음.
2. 함수에서 두개 이상의 값을 반환해야 할 때 사용할 수 있음.
3. 큰 데이터 구조를 간단한 방법으로 참조할 수 있음.

메모리 특징

- 메모리는 바이트 단위로 접근할 수 있음.
- 첫번째 바이트의 주소는 n , 두번째 바이트는 $n+1$, ...

변수와 메모리

- 변수의 자료형에 따라서 차지하는 메모리 공간이 달라짐.
- char형 변수: 1바이트
- int형 변수: 4바이트.


[참조 연산자와 주소 연산자]

참조 연산자 (*)를 이용하여 포인터 변수가 가지고 있는 값이 *가르키는 메모리 주소의 값을 읽거나 수정*할 수 있음.

주소 연산자(&)를 이용하여 *변수의 주소 값을 얻을 수 있다.*

배열 (Array)

배열 : 동일한 타입의 데이터가 여러 개 저장 되어 있는 데이터 저장 장소.

 Why do we need an array

1. 프로그램의 가독성.
2. 효율적인 메모리 관리.
3. 유지보수가 쉽다.

예시)

학생 100명의 값을 저장하는 개별 변수를 선언해야 했음.

```
int student1;
int student2;
int student3;
.
.
.
int student100;
```

그러나, 배열을 사용하면 학생 100명의 정보를 하나의 변수(배열)에서 저장하고 관리할 수 있음.

```
int student[100];
```

배열 선언 방법

```
(자료형) (배열명) ([배열의 크기]);
int student[100];
```

 Important

1. 배열의 index는 항상 0부터 시작함.
2. Index가 배열의크기를벗어나지않는다.
3. Index 값은0과 양의정수로이루어진다.

함수(Function)

함수 : 특정 기능을 하도록 만든 코드의 집합.

함수는 반드시 **호출에 의해서** 함수 내부에 있는 내용이 실행됨.

C언어에서 함수는 크게 두 가지로 나뉨.

- 라이브러리 함수 : printf(), scanf(), strcmp() 등.
- 사용자 지정 함수 : GettingNumber(), Calculator() 등.

Advantage of function

1. 코드의 재사용 가능 => 효율적.
2. 가독성 증가.
3. 유지관리 용이.

함수 선언 방법

```
(Type) (name)((Parameter));  
int AddingNumber(int n1, int n2);
```

함수 정의

```
int AddingNumber(int n1, int n2){  
    int result = 0;  
    result = n1 + n2;  
    return result; // or return n1 + n2;  
}
```

Important

매개변수(Parameter)와 인자, 인수(Argument)는 다른 개념임.

매개변수 : 함수 선언 시 변수 목록.

```
int AddingNumber(int n1, int n2);
```

위 코드에서 n1과 n2를 매개변수라고 함.

인자 = 인수(Argument) : 함수가 호출될 때 전달되는 실제 값.

```
result = AddingNumber(10, 20);
```

위 코드에서 10과 20을 인자, 인수(Argument)라고 함.

과제1-2

과제 설명

15쪽 *각 함수별 작동원리* 정리해서 제출(종이에 그림을 그려 스캔 후 제출, ppt로 만들어서 제출.. 등 제출 양식은 자유)

[소스코드]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

void myPrint1(int* a, int row, int col);
void myPrint2(int a[4][3], int row, int col);
void myPrint3(int a[][3], int row, int col);
void myPrint4(int (*a)[3], int row, int col);

int main() {

    int arr[4][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12} };
    int row = sizeof(arr) / sizeof(arr[0]);
    int col = sizeof(arr[0]) / sizeof(arr[0][0]);

    myPrint1(arr, row, col);
    myPrint2(arr, row, col);
    myPrint3(arr, row, col);
    myPrint4(arr, row, col);

    return 0;
}

void myPrint1(int* a, int row, int col) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) printf("%d ", *(a + i * col + j));
    }
    printf("\n");
}

void myPrint2(int a[4][3], int row, int col) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) printf("%d ", a[i][j]);
    }
}
```

```

    }
    printf("\n");
}
void myPrint3(int a[][3], int row, int col) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) printf("%d ", a[i][j]);
    }
    printf("\n");
}
void myPrint4(int (*a)[3], int row, int col) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) printf("%d ", *(*a+i) + j);
    }
    printf("\n");
}
}

```

[코드 풀이]

전처리문

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

```

1. scanf() 함수를 Visual Studio 상에서 사용하기 위해 Secure 관련 정의를 하였습니다.
2. C파일에서 사용할 표준 라이브러리 함수를 포함 시킵니다.

Tip

Stdio 헤더 파일은 Standard Input/Output library로서, printf() 함수와 scanf() 등의 함수가 내장되어 있습니다.

C프로그래밍을 공부할 때, 사용할 헤더 파일에 어떤 함수들이 있는지 찾아보는 습관이 중요합니다.

함수 프로토타입 선언하기

```

void myPrint1(int* a, int row, int col);
void myPrint2(int a[4][3], int row, int col);
void myPrint3(int a[][3], int row, int col);
void myPrint4(int (*a)[3], int row, int col);

```

3. 본 프로그램에서 사용할 사용자 지정 함수의 원형(Prototype)을 선언하였습니다.
함수 원형을 선언하게 되면, 선언한 사용자 지정 함수에 대한 디자인을 main() 함수 이후에 작성하여 코드의 가독성을 높일 수 있습니다.

Main함수 내부 작동하기


```

int main() {

    int arr[4][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12} };
    int row = sizeof(arr) / sizeof(arr[0]);
    int col = sizeof(arr[0]) / sizeof(arr[0][0]);

    myPrint1(arr, row, col);
    myPrint2(arr, row, col);
    myPrint3(arr, row, col);
    myPrint4(arr, row, col);

    return 0;
}

```

4. int Type의 2차원 배열 **arr**을 선언합니다.
5. **arr**의 요소에는 1부터 ~ 12까지 포함되어 있습니다.
6. int Type의 row 변수를 선언하였으며, sizeof(arr) 값(48)에 sizeof(arr[0]) 값(12)을 나누어 할당합니다.
 sizeof(arr) = arr의 전체 byte 수입니다.
 arr은 int type이며 총 12개의 요소가 포함되어 있습니다.
 각 요소는 4byte이며, 12개의 요소를 포함한 arr은 $4 * 12 = 48\text{byte}$ 를 확보하고 있는 상태입니다.
 sizeof(arr[0]) = arr[0]의 전체 byte 수입니다.
 2차원 배열 arr의 0번째 index는 {1, 2, 3}입니다.
 arr[0] = {1, 2, 3}로 세 개의 요소를 할당 받고 있습니다.
 arr[0]의 byte수는 int type * arr[0]의 요소의 개수 = $4 * 3 = 12\text{byte}$ 가 됩니다.
 즉, row에는 $48 / 12 = 4\text{byte}$ 가 할당 되었습니다.
7. int Type의 col 변수를 선언하였으며, sizeof(arr[0]) 값(12)에 sizeof(arr[0][0]) 값(4)를 나누어 할당합니다.
8. myPrint1() 함수를 호출하며, argument로 arr, row, col을 전달합니다.
14. myPrint1() 함수가 종료되면, myPrint2() 함수를 호출하며, argument로 arr, row, col을 전달합니다.
20. myPrint2() 함수가 종료되면, myPrint3() 함수를 호출하며, argument로 arr, row, col을 전달합니다.
26. myPrint3() 함수가 종료되면, myPrint4() 함수를 호출하며, argument로 arr, row, col을 전달합니다.
32. myPrint4() 함수가 종료되면, return 0을 통해 프로그램이 종료됩니다.

Myprint1()

```

void myPrint1(int* a, int row, int col) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) printf("%d ", *(a + i * col + j));
    }
    printf("\n");
}

```

9. myPrint1() 함수의 parameter 값으로 arr, row, col을 전달 받았습니다.
 int Type의 포인터 변수 a는 전달 받은 배열 arr를 참조하고 있습니다. // *a = arr
 int Type의 변수 row는 전달 받은 변수 row가 대입 되었습니다. // row = 4
 int Type의 변수 col은 전달 받은 변수 col이 대입 되었습니다. // col = 3
10. i가 0부터 ~ row 값 미만일 때까지 첫 번째 for문을 반복 합니다. // 첫 번째 for문은 네 번 반복.
11. j가 0부터 ~ col 값 미만일 때까지 두 번째 for문을 반복합니다. // 두 번째 for문은 세 번 반복.
12. 두 번째 for문 내부에서 포인터 변수 a의 (i * col + j) 값을 참조하여, 참조된 주소의 값을 10진수 형태로 출력합니다.

계산 과정

- a. $*(a + 0 * 4 + 0) \Rightarrow *(a)$ 가 가리키는 주소의 값을 출력함. -> '1'이 출력됨.
- b. $*(a + 0 * 4 + 1) \Rightarrow *(a+1)$ 가 가리키는 주소의 값을 출력함. -> '2'가 출력됨.
- .
- .
- .
- l. $*(a + 2 * 4 + 3) \Rightarrow *(a+11)$ 가 가리키는 주소의 값을 출력함. -> '12'가 출력됨.

13. 개행 후, myPrint1() 함수를 종료하고, main함수의 myPrint1()이 호출된 이후로 돌아갑니다.

Myprint2()

```
void myPrint2(int a[4][3], int row, int col) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) printf("%d ", a[i][j]);
    }
    printf("\n");
}
```

15. myPrint2() 함수의 parameter 값으로 arr, row, col을 전달 받았습니다.
 int Type의 2차원 배열 a는 4 * 3 사이즈이며, 전달 받은 배열 arr를 참조하고 있습니다. // a = arr
 int Type의 변수 row는 전달 받은 변수 row가 대입 되었습니다. // row = 4
 int Type의 변수 col은 전달 받은 변수 col이 대입 되었습니다. // col = 3
16. i가 0부터 ~ row 값 미만일 때까지 첫 번째 for문을 반복 합니다. // 첫 번째 for문은 네 번 반복.
17. j가 0부터 ~ col 값 미만일 때까지 두 번째 for문을 반복합니다. // 두 번째 for문은 세 번 반복.
18. 두 번째 for문 내부에서 2차원 변수 a의 [i][j] 값을 참조하여, 참조된 주소의 값을 10진수 형태로 출력합니다.

계산 과정

- a. $a[0][0] \Rightarrow a$ 가 가리키는 0번째 주소의 값을 출력함. -> '1'이 출력됨.
- b. $a[0][1] \Rightarrow a + 1$ 이 가리키는 주소의 값을 출력함. -> '2'가 출력됨.
- .
- .

.
l. `a[4][3]` => `a + 11`가 가리키는 주소의 값을 출력함. -> '12'가 출력됨.

19. 개행 후, `myPrint2()` 함수를 종료하고, `main`함수의 `myPrint2()`이 호출된 이후로 돌아갑니다.

Important

Q. 2차원 배열은 어떻게 메모리에 저장이 되는가?

A. 2차원 배열은 메모리에 일직선으로 저장이 됩니다.

2차원 배열 `arr[4][3]`을 선언하고, 2차원 배열 `arr[0][0]`은 배열 `arr`가 참조하고 있는 첫 번째 주소를 가리킵니다. // `arr[0][0] = arr + 0` 혹은 `arr[0][0] = arr`.

`arr[0][1]`은 배열 `arr`가 참조하고 있는 두 번째 주소를 가리킵니다. // `arr[0][1] = arr + 1`.

보이기 편하도록 2차원 배열(행렬, 테이블)로 디자인하지만, 실제로는 메모리에 일직선(=일차원)으로 저장되기에 본 개념을 알고 있으면 배열과 포인터를 사용하는 데 있어 수월합니다.

Myprint3()

```
void myPrint3(int a[][3], int row, int col) {  
    for (int i = 0; i < row; i++) {  
        for (int j = 0; j < col; j++) printf("%d ", a[i][j]);  
    }  
    printf("\n");  
}
```

21. `myPrint3()` 함수의 parameter 값으로 `arr`, `row`, `col`을 전달 받았습니다.

`int` Type의 2차원 배열 `a`는 `[] * 3` 사이즈이며, 전달 받은 배열 `arr`를 참조하고 있습니다. // `a = arr`

`int` Type의 변수 `row`는 전달 받은 변수 `row`가 대입 되었습니다. // `row = 4`

`int` Type의 변수 `col`은 전달 받은 변수 `col`이 대입 되었습니다. // `col = 3`

22. `i`가 0부터 ~ `row` 값 미만일 때까지 첫 번째 for문을 반복 합니다. // 첫 번째 for문은 네 번 반복.

23. `j`가 0부터 ~ `col` 값 미만일 때까지 두 번째 for문을 반복합니다. // 두 번째 for문은 세 번 반복.

24. 두 번째 for문 내부에서 2차원 변수 `a`의 `[i][j]` 값을 참조하여, 참조된 주소의 값을 10진수 형태로 출력합니다.

계산 과정

a. `a[0][0]` => `a`가 가리키는 0번째 주소의 값을 출력함. -> '1'이 출력됨.

b. `a[0][1]` => `a + 1`이 가리키는 주소의 값을 출력함. -> '2'가 출력됨.

.
.
.

l. `a[4][3]` => `a + 11`가 가리키는 주소의 값을 출력함. -> '12'가 출력됨.

25. 개행 후, `myPrint3()` 함수를 종료하고, `main`함수의 `myPrint3()`이 호출된 이후로 돌아갑니다.

Important

Q. `int a[][3]`; 로 parameter를 선언한 이유는 무엇인가?

A. 2차원 배열을 선언할 때, 행에 대한 index는 생략해도 되지만, 열에 대한 index는 무조건 포함되어야 합니다.

열에 대한 값이 index로 지정되어 있으면, 행은 2차원 배열의 전체 요소를 열로 나누어 자동으로 배정합니다.

예시)

```
int a[][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}};
```

위 코드에서 2차원 배열 a는 총 12개의 요소를 가지고 있습니다.

배열 a를 선언할 때 행에 대한 index는 생략하고, 열에 대한 index만 3으로 지정하였습니다.

C언어는 열이 3인 것을 파악하여, a의 전체 요소에서 3을 나눈 값(4)을 행에 자동으로 할당합니다.

정리하자면, C언어 컴파일러가 열의 크기에 따라 자동으로 행 값을 할당하므로 행에 대한 index는 생략해도 된다.

단, 열에 대한 index는 절대로 생략하면 안 된다.

Myprint4()

```
void myPrint4(int (*a)[3], int row, int col) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) printf("%d ", *(a+i) + j));
    }
    printf("\n");
}
```

27. myPrint4() 함수의 parameter 값으로 arr, row, col을 전달 받았습니다.

int Type의 배열 포인터(=2차원 포인터) a는 행을 3으로 합니다. // $*(a + 0) = arr[0]$, $*(a + 1) = arr[1]$, $*(a + 2) = arr[2]$.

int Type의 변수 row는 전달 받은 변수 row가 대입 되었습니다. // row = 4

int Type의 변수 col은 전달 받은 변수 col이 대입 되었습니다. // col = 3

28. i가 0부터 ~ row 값 미만일 때까지 첫 번째 for문을 반복 합니다. // 첫 번째 for문은 네 번 반복.

29. j가 0부터 ~ col 값 미만일 때까지 두 번째 for문을 반복합니다. // 두 번째 for문은 세 번 반복.

30. 두 번째 for문 내부에서 2차원 포인터 a 주소 내부에 j가 참조하고 있는 주소의 값을 참조하여, 참조된 주소의 값을 10진수 형태로 출력합니다.

계산 과정

a. $*(a + 0) + 0 \Rightarrow *(a + 0)$ 이 가리키는 주소에 0을 더한 주소의 값을 출력함. -> '1'이 출력됨.

b. $*(a + 0) + 1 \Rightarrow *(a + 0)$ 이 가리키는 주소에 1을 더한 주소의 값을 출력함. -> '2'가 출력됨.

.

.

l. $*(a + 3) + 2 \Rightarrow *(a + 3)$ 이 가리키는 주소에 2를 더한 주소의 값을 출력함. -> '12'가 출력됨.

31. 개행 후, myPrint3() 함수를 종료하고, main함수의 myPrint4()가 호출된 이후로 돌아갑니다.

Important

Q. $(*a)[3]$ 은 무엇인가?

A. $(*a)[3]$ 은 **2차원 포인터**이며, **배열 포인터**라고 불리기도 합니다.

$(*a)[3]$ 을 풀어서 작성하면 다음과 같습니다.

```
int arr[4][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}};  
int (*a)[3] = arr;
```

2차원 포인터 a는 arr를 가리킵니다.

$(*a) = arr[0]$. // 주소를 가리킴.

$(*a + 1) = arr[1]$.

$(*a + 2) = arr[2]$.

$(*a + 3) = arr[3]$.

a는 2차원 배열 arr의 0번째부터 3번째 index까지 참조합니다.

여기서 a는 2차원 포인터이므로, arr의 0번째 요소가 가리키는 {1, 2, 3}의 주소에서 각 요소에도 접근을 할 수 있습니다.

$*(a + 0) + 0$ 이는 $(a + 0)$ 이 가리키는 주소 ($arr[0]$)에서, $((a + 0) + 0)$ 은 $arr[0][0]$ 을 참조합니다.

중첩 참조된 주소의 값을 확인하기 위해서는 에스터리스크(*)를 두 번 사용하여 확인할 수 있습니다.

과제1-3

과제 설명

25쪽 **이중포인터가 포함된 함수의 작동원리**에 대해 서술하시오.(자유양식)

[소스 코드]

```
#define _CRT_SECURE_NO_WARNINGS  
#include <stdio.h>  
  
void SetStr(char** str);  
  
int main() {  
  
    char* str;  
  
    str = "I'am sad";  

```

```

        SetStr(&str);

        printf("%s", str);

        return 0;
    }

    void SetStr(char** str) {
        *str = "I'am happy";
    }

```

[코드 풀이]

전처리문

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

```

1. scanf() 함수를 Visual Studio 상에서 사용하기 위해 Secure 관련 정의를 하였습니다.
2. C파일에서 사용할 표준 라이브러리 함수를 포함 시킵니다.

함수 프로토타입 선언하기

```

void SetStr(char** str);

```

3. 본 프로그램에서 사용할 사용자 지정 함수의 원형(Prototype)을 선언하였습니다.
본 함수의 type은 void Type으로, return을 하지 않습니다.

Main함수 내부 작동하기

```

int main() {

    char* str;

    str = "I'am sad";

    SetStr(&str);

    printf("%s", str);

    return 0;
}

```

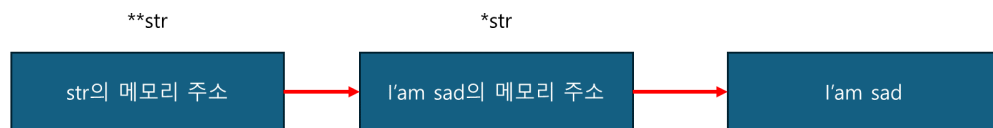
4. char Type의 포인터 변수 str를 선언합니다.
5. str은 "I'm sad"라는 문자열이 담긴 주소를 가리킵니다.
6. SetStr() 함수를 호출하며, argument 값으로 포인터 변수 str을 전달합니다.
10. SetStr() 함수를 통해 변경된 str이 가리키는 주소를 출력합니다. // "I'am happy"의 주소를 가리키고 있으므로, "I'am Happy"가 출력 됨.
11. 프로그램을 종료합니다.

Setstr()

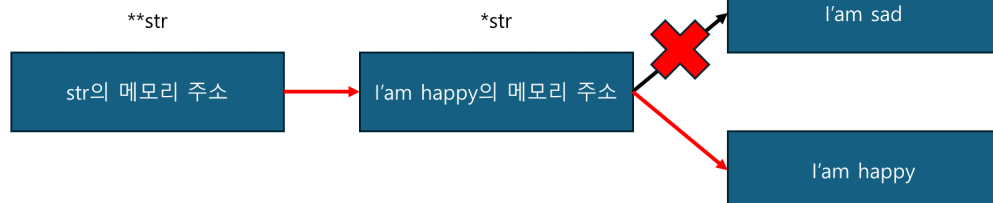
```
void SetStr(char** str) {
    *str = "I'am happy";
}
```

7. void Type의 함수 SetStr()은 parameter로 str의 메모리 주소를 전달 받았습니다. // SetStr의 변수 str은 parameter로 전달 받은 str의 주소를 참조합니다.
이중 포인터 str = &str(전달 받은 포인터의 주소).
[포인터 참조 정리]
main() 함수에서 포인터 str은 "I'm sad"라는 문자열의 주소를 가리키고 있음.
str의 주소("I'm sad")를 SetStr() 함수의 argument로 전달함.
SetStr() 함수에서 parameter로 받은 str의 주소는 이중포인터 str이 가리킴.
str(SetStr) -> &str(main) -> "I'm Sad".
8. SetStr() 함수 내부의 *str은 parameter로 받은 main() 함수의 포인터 str을 가리키며, 이 str이 "I'am happy"의 주소를 참조하도록 합니다.
9. SetStr() 함수를 종료하고, main함수의 SetStr()가 호출된 이후로 돌아갑니다.

*str = "I'am happy" 로 변경하기 이전



*str = "I'am happy" 로 변경하기 이후



HW_003

과제 설명

조건 : void print(int *row, int *col, int **pArr) 함수 사용

void arr_ij(int *sizeRow, int *sizeCol, int **pArr) 함수 사용 (달팽이 만드는 함수)

과제 3



```
int main()
{
    int **arr = NULL;
    int row,col,sizeRow,sizeCol;

    printf("열의 수를 입력하세요:");
    scanf("%d",&sizeCol);
    printf("행의 수를 입력하세요:");
    scanf("%d",&sizeRow);

    row = sizeRow;
    col = sizeCol;
```

2차원 동적 메모리 할당 필요

```
arr_ij(&sizeRow,&sizeCol,arr);

print(&row,&col,arr);

for(int i=0; i<row; i++){
    free(arr[i]);
}

return 0;
}
```

조건 :

void print(int *row, int *col, int **pArr) 함수 사용

void arr_ij(int *sizeRow, int *sizeCol, int **pArr) 함수 사용
(달팽이 만드는 함수)

```
열의 수를 입력하세요:10
행의 수를 입력하세요:10
1  2  3  4  5  6  7  8  9 10
36 37 38 39 40 41 42 43 44 11
35 64 65 66 67 68 69 70 45 12
34 63 84 85 86 87 88 71 46 13
33 62 83 96 97 98 89 72 47 14
32 61 82 95 100 99 90 73 48 15
31 60 81 94 93 92 91 74 49 16
30 59 80 79 78 77 76 75 50 17
29 58 57 56 55 54 53 52 51 18
28 27 26 25 24 23 22 21 20 19
```

[머리말]

본 과제는 제한 시간 내에 끝까지 수행해 내지 못했습니다. 달팽이 알고리즘을 과제 제출 이후 개별적으로 공부하여 다시 시도해 보겠습니다.

[소스코드]

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

void print(int* row, int* col, int** pArr);
void arr_ij(int* sizeRow, int* sizeCol, int** pArr);

int main() {

    // 더블포인터 변수 및 일반 변수 선언하기.
    int** arr = NULL;
    int row, col, sizeRow, sizeCol;
```



```

// 입출력 받기.
printf("열의 수를 입력하세요:");
scanf("%d", &sizeCol);
printf("행의 수를 입력하세요:");
scanf("%d", &sizeRow);

// 변수 할당하기.
row = sizeRow;
col = sizeCol;

// 2차원 동적 메모리 할당하기.
arr_ij(&sizeRow, &sizeCol, arr);

if (arr == NULL) printf("동적 할당 되지 않았습니다.");
else if (arr != NULL) printf("정상적으로 동적 할당 되었습니다.");
// 배열 arr 출력하기.
/*print(&row, &col, &arr);*/

// 동적 메모리 해제하기.
for (int i = 0; i < row; i++) {
    free(arr[i]);
}

return 0;
}

// 달팽이 형식에 맞게 출력하는 함수.
void print(int* row, int* col, int** pArr) {

    pArr = (int**)malloc(sizeof(int) * (*row));

    /*printf("%d %d %d", *row, *col, sizeof(*(pArr + 0) + 1));*/
    printf("%d %d %d", *row, *col, sizeof(*(pArr + 0)));

    for (int i = 0; i < *row; i++) {
        for (int j = 0; j < *col; j++) {
            printf("%3d ", (*(pArr + i) + j));
        }
        printf("\n");
    }
}

// 달팽이 형식에 맞게 구현하는 함수.
void arr_ij(int* sizeRow, int* sizeCol, int** pArr) {

    int inputNum = 1; // 배열에 넣을 숫자 변수.
    int change = 0; // change가 10이 될 때마다 행과 열의 번호를 삼입함.
    // 2차원 배열 만들기.
    // 1, sizeRow 만큼의 포인터 배열을 동적 할당함.
    pArr = (int**)malloc(sizeof(int) * (*sizeRow));
    if (pArr == NULL) printf("동적할당 실패함.\n");
    for (int i = 0; i < (*sizeRow); i++) {

```

```

// 2. 각각의 포인터 배열 요소에 sizeCol 만큼의 포인터 배열을 동적 할당함.
*(pArr + i) = (int*)malloc(sizeof(int) * (*sizeCol));
}

int cntRow = 0, cntCol = 0;
int inputNumLast = 0; // 마지막 숫자 저장용 (행, 열). inputNum과는 다름.
int head = 1; // 1 : 오른쪽, 2 : 아래, 3 : 왼쪽, 4 : 위로.
int t1 = 0, t2 = 0, t3 = 0, t4 = 0;

for (int i = 0; i < (*sizeRow); i++) {
    change = 0;

    for (int j = 0; j < (*sizeCol); j++) {
        int tempInputNum = inputNum;
        // 최초 1회 inputNumLast 지정하기.
        if (i == 0 && head == 1) {
            inputNumLast = inputNum;
            t1++;
        }

        if (((0 < i && i < 9) && j == (*sizeCol) - 1) && head == 2) { //
            inputNumLast++;
            inputNum = inputNumLast;
            (*(pArr + i) + j) = inputNum;
            t2++;
        } else if ((i == (*sizeRow) - 1 && j < ((*sizeCol))) && head ==
3) {
            inputNumLast++;
            inputNum = inputNumLast;
            (*(pArr + i) + (*sizeCol) - 1 - j) = inputNum;
            t3++;
        } else if ((j == 0 && (0 < i && i < 9)) && head == 4) {
            inputNumLast++;
            inputNum = inputNumLast;
            (*(pArr + i) + j) = 0;
            t4++;
        } else {
            (*(pArr + i) + j) = inputNum;
        }

        inputNum = tempInputNum;

        // 변수 증감하기.
        change++;
        inputNum++;
    }
    cntRow++;
    if (t1 == (*sizeCol) - 2 || t1 == (*sizeCol)) {
        /*printf("t1 : %d\n\n\n", t1);*/
        t1 = 0;
        t2 = 0;
    }
}

```

```

        t3 = 0;
        t4 = 0;
        head = 2;
    }else if (t2 == (*sizeCol) - 2) {
        /*printf("t2 : %d\n\n\n\n", t2);*/
        t1 = 0;
        t2 = 0;
        t3 = 0;
        t4 = 0;
        head = 3;
    }else if (t3 == (*sizeRow) - 2 || t3 == (*sizeCol)) {
        /*printf("t3 : %d\n\n\n\n", t3);*/
        t1 = 0;
        t2 = 0;
        t3 = 0;
        t4 = 0;
        head = 4;
    }else if (t4 == (*sizeCol) - 3) {
        /*printf("t4 : %d\n\n\n\n", t4);*/
        t1 = 0;
        t2 = 0;
        t3 = 0;
        t4 = 0;
        head = 1;
    }
    /*printf("t4 : %d\n\n\n\n", t4);*/

}

for (int i = 0; i < (*sizeRow); i++) {
    for (int j = 0; j < (*sizeCol); j++) {
        printf("%3d ", (*(pArr + i) + j));
        /*printf("%3d[%d][%d] ", (*(pArr + i) + j), i, j);*/
    }
    printf("\n");
}
printf("\n\n%d\n\n", (*(pArr + 0) + 9));
/*print(sizeRow, sizeCol, pArr);*/

}

```

[실행 결과]
Test Case#1

```
Microsoft Visual Studio 디버그
열의 수를 입력하세요:10
행의 수를 입력하세요:10
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 11
21 22 23 24 25 26 27 28 29 12
31 32 33 34 35 36 37 38 39 13
41 42 43 44 45 46 47 48 49 14
51 52 53 54 55 56 57 58 59 15
61 62 63 64 65 66 67 68 69 16
71 72 73 74 75 76 77 78 79 17
81 82 83 84 85 86 87 88 89 18
28 27 26 25 24 23 22 21 20 19

10

동적 할당 되지 않았습니다.
C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\Homework\HW_20240709\x64\Debug\HW_20240709.exe
(프로세스 29568개)이(가) 종료되었습니다(코드: -1073741819개).
이 창을 닫으려면 아무 키나 누르세요...|
```

Ideas & Important Information

 Info

아이디어 및 중요 정보를 작성합니다.

Memo

 Info

작업 중 기타 내용을 메모합니다.

공책에다가 메모하였음.

Review

 Info

하루 작업에 대한 피로도, 기분 등을 평가합니다.

Feelings : (😓)

Fatigue : (4)

Summary : ~~~