ROBIT
ROBOT SPORT GAME TEAM

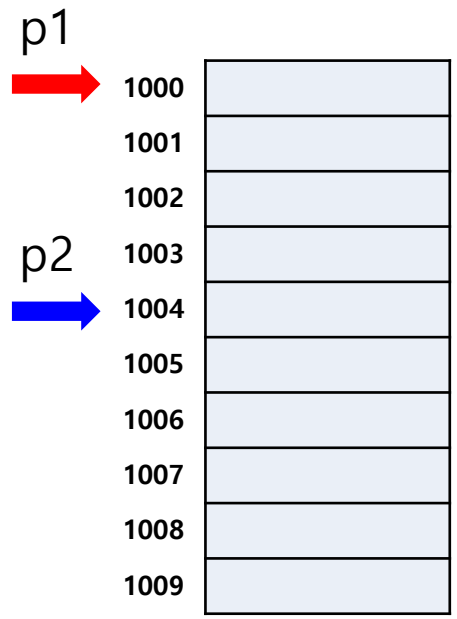
C Language – Pointer2

포인터 산술의 제한적 활용

- 포인터 변수에 offset값을 더하거나 뺄 수 있다.
  - p1 + 3 : p1 이후 3번째 요소의 시작주소

- 타입이 같은 포인터끼리는 뺄 수 있다.
  - ex) p2 – p1 : p1과 p2 사이에 몇 개의 배열 요소가 존재하는지
  - 두 개의 포인터 변수를 더할 수는 없다

- 산술연산자 '*', '/', '%' 는 포인터 변수를 피연산자로 취할 수 없다.



배열과 함수

ROBIT
ROBOT SPORT GAME TEAM

일반적으로 함수의 인자로 배열을 넘길 때, call by reference가 사용된다
즉, 함수의 인자로 배열의 시작 주소값을 넘긴다.

배열과 함수

ROBIT
ROBOT SPORT GAME TEAM

- 배열 형태로 선언된 매개변수에 1차원 배열의 이름이 나타내는 시작 주소값 전달

선언

```
void Function(int array[], int n)
{
  …
}
```

호출

```
int main(void)
{
  int array[5];
  Function(array, 5);
  Function(&array, 5);
  Function(&array[0], 5);
}
```

배열과 함수

ROBIT
ROBOT SPORT GAME TEAM

- 포인터 변수에 1차원 배열의 시작 주소값 전달

선언

```
void Function(int* array, int n)
{
  …
}
```

호출

```
int main(void)
{
  int array[5];
  Function(array, 5);
  Function(&array, 5);
  Function(&array[0], 5);
}
```

배열과 함수

- 매개변수로 배열의 길이를 같이 전달해,
  초과하는 인덱스로의 접근을 방지해야 한다!!

선언

```
void Function(int* array, int n)
{
  …
}
```

호출

```
int main(void)
{
  int array[5];
  Function(array, 5);
  Function(&array, 5);
  Function(&array[0], 5);
}
```

배열과 함수

ROBIT
ROBOT SPORT GAME TEAM

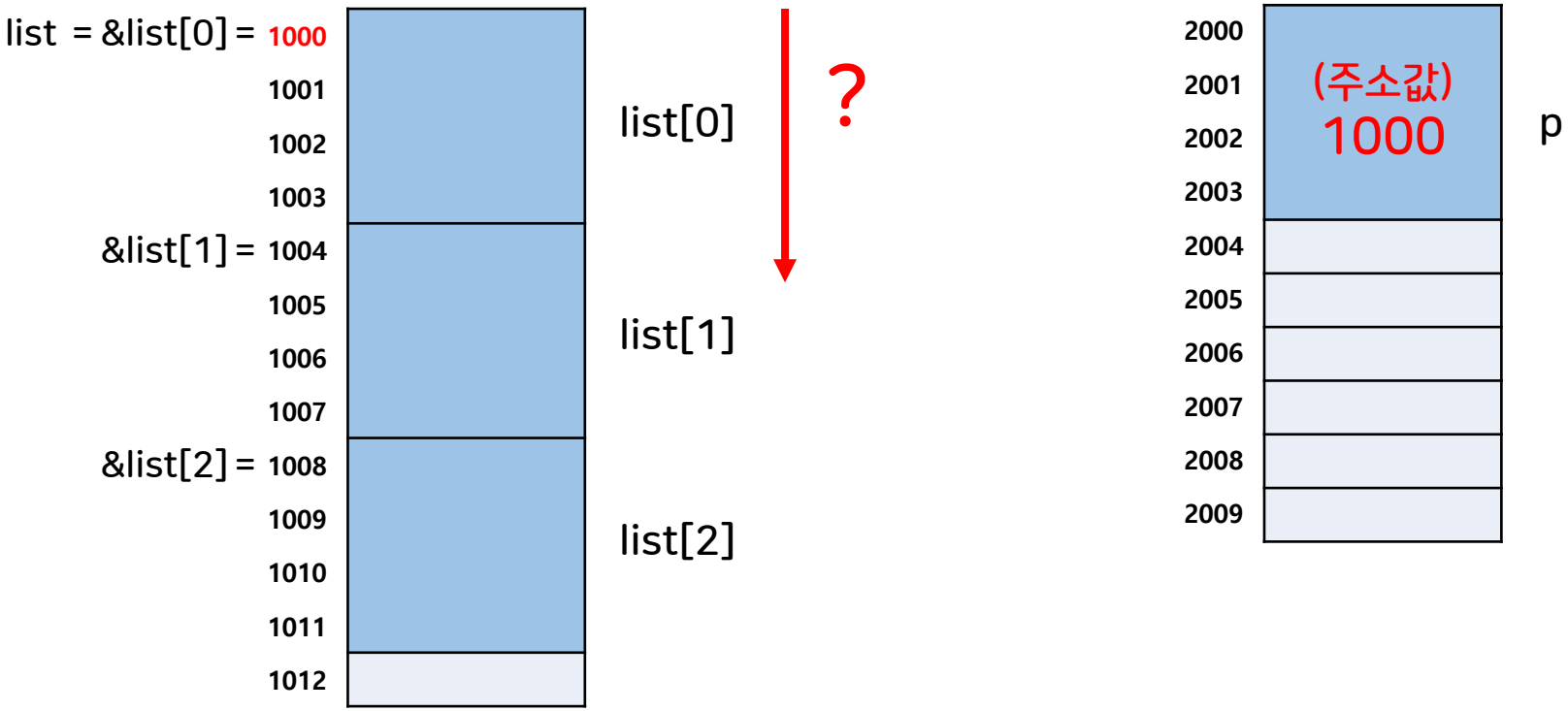
굳이 매개변수로 배열의 길이를 전달하지 않아도 컴파일하고 실행하는 데에는 문제가 없지만…

```c
#include <stdio.h>

void function1(int list[]);

int main(void)
{
    int list[5] = { 10, 20, 30, 40, 50 };

    function1(&list);

    for (int i = 0; i < 5; i++)
    {
        printf("%d ", list[i]);
    }
    printf("₩n");

    return 0;
}

void function1(int list[])
{
    list[0] = 1;
}
```

```
1 20 30 40 50
```

배열과 함수

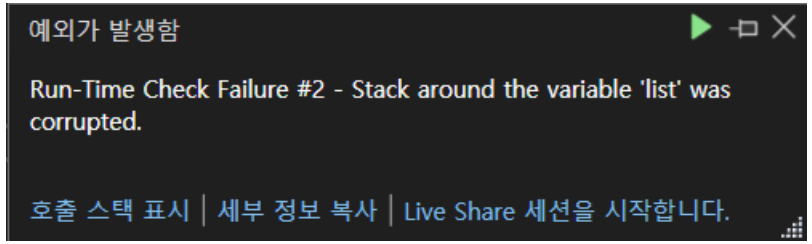
'함수'의 입장해서 생각해 보면, 배열의 시작주소만을 받았을 뿐,
어디가 끝인지 알 수 없다.



배열과 함수

따라서 다음과 같이 기존 배열의 크기가 넘는 인덱스에 접근을 시도하면, 컴파일 오류는 나지 않지만, 실행 할 때 문제가 생긴다.

```c
#include <stdio.h>

void function1(int list[]);

int main(void)
{
    int list[5] = { 10, 20, 30, 40, 50 };

    function1(&list);

    for (int i = 0; i < 5; i++)
    {
        printf("%d ", list[i]);
    }
    printf("₩n");

    return 0;
}

void function1(int list[])
{
    list[100] = 1;
}
```



주로 배열의 인덱스를 벗어나는 잘못된 메모리 접근으로 발생하는 에러

배열과 함수

다음과 같이 배열의 크기를 전달하여, 함수가 배열의 최대 인덱스를 알 수 있도록 하자

```c
#include <stdio.h>

void function1(int list[], int size);

int main(void)
{
    int list[5] = { 10, 20, 30, 40, 50 };

    function1(&list, 5);

    return 0;
}

void function1(int list[], int size)
{
    for (int i = 0; i < size; i++)
    {
        printf("%d ", list[i]);
    }
    printf("₩n");
}
```

```
10 20 30 40 50
```

sizeof 연산자

ROBIT
ROBOT SPORT GAME TEAM

sizeof() : 피연산자의 크기를 byte 단위로 반환하는 함수.
ex) 배열, 자료형, 변수 등등

```c
#include <stdio.h>

int main(void)
{
    int list1D[5] = { 10, 20, 30, 40, 50 };
    int list2D[4][3] = { {1,  2,  3},  {4,  5,  6},
                         {7,  8,  9},  {10, 11, 12}};

    size_t size;

    size = sizeof(list1D);
    printf("size : %d₩n", size);

    size = sizeof(list1D[0]);
    printf("size : %d₩n", size);

    size = sizeof(list2D);
    printf("size : %d₩n", size);

    size = sizeof(list2D[0]);
    printf("size : %d₩n", size);

    size = sizeof(list2D[0][0]);
    printf("size : %d₩n", size);

    size = sizeof(double);
    printf("size : %d₩n", size);

    return 0;
}
```

```
size : 20
size : 4
size : 48
size : 12
size : 4
size : 8
```

size_t 자료형

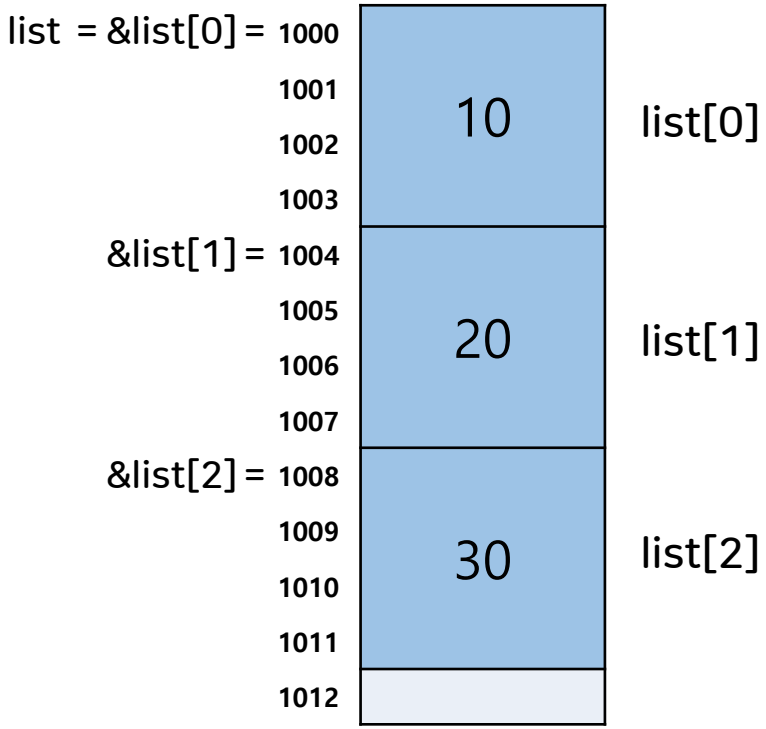
ROBIT
ROBOT SPORT GAME TEAM

size_t : 부호없는 정수형(unsigned int)의 자료형
int와 마찬가지로 4byte의 크기를 가진다

배열의 크기

sizeof() 연산자를 활용하여 배열의 크기를 구할 수 있다.

```
int main(void)
{
  int i[3] = {10,20,30};
}
```



배열의 전체크기 : 12
배열의 한칸 크기(int) : 4

배열의 크기 → 12 / 4 = 3

배열의 크기

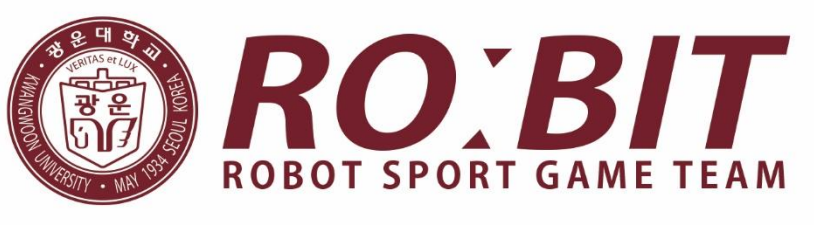
ROBIT
ROBOT SPORT GAME TEAM

sizeof() 연산자를 활용하여 배열의 크기를 구할 수 있다.

```c
#include <stdio.h>

void function(int* list, size_t size);

int main(void)
{
    int list[5] = { 10, 20, 30, 40, 50 };

    size_t size;
    size = sizeof(list) / sizeof(int);
    function(list, size);

    size = sizeof(list) / sizeof(list[0]);
    function(list, size);

    return 0;
}

void function(int* list, size_t size)
{
    for (size_t i = 0; i < size; i++)
    {
        printf("%d ", *(list+i));
    }
    printf("₩n");
}
```

```
10 20 30 40 50
10 20 30 40 50
```

2차원 배열과 함수



다양한 방법으로 2차원 배열을 함수에서 참조할 수 있다.

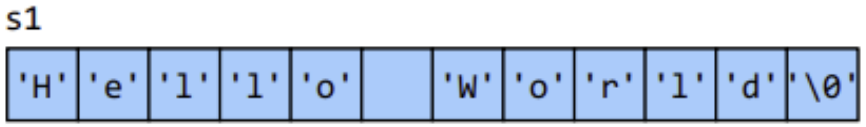
```c
#include<stdio.h>

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);
}
```

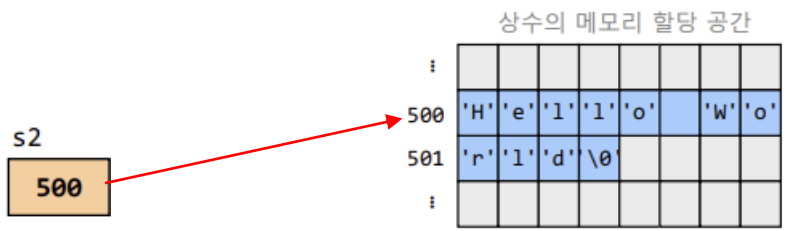
```c
void myPrint1(int* a, int row, int col) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) printf("%d ", *(a + i * col + j));
        printf("₩n");
    }
}

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");
    }
}
```

포인터와 문자열

- 배열 형태로 선언된 매개변수에 1차원 배열의 이름이 나타내는 시작 주소값 전달

```
int main(void)
{
  char s1[] = "Hello World";
  char* s2 = "Hello World";
}
```





포인터와 문자열

ROBIT
ROBOT SPORT GAME TEAM

- 배열 형태로 선언된 매개변수에 1차원 배열의 이름이 나타내는 시작 주소값 전달

```c
#include <stdio.h>

int main(void)
{
    char s1[] = "Hello World";
    char* s2 = "Hello World";

    s1[0] = 'h';     // 가능
    // s2[0] = 'h';  // 불가
```

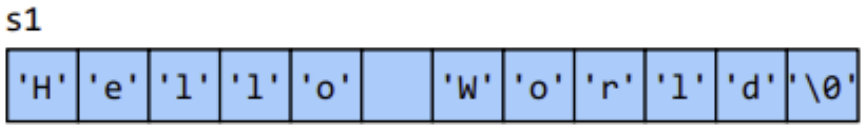
s1의 경우 개별 요소값 수정 가능

s2의 경우 개별 요소값 수정 불가

포인터와 문자열

- 배열 형태로 선언된 매개변수에 1차원 배열의 이름이 나타내는 시작 주소값 전달

```c
#include <stdio.h>

int main(void)
{
    char s1[] = "Hello World";
    char* s2 = "Hello World";

    s1[0] = 'h';     // 가능
    // s2[0] = 'h';  // 불가
```



s1의 경우 처음 초기화된 배열의 크기 이상의 작업을 할 수 없다는 단점이 있다.

포인터와 문자열

ROBIT
ROBOT SPORT GAME TEAM

- 배열 형태로 선언된 매개변수에 1차원 배열의 이름이 나타내는 시작 주소값 전달

```c
#include <stdio.h>

int main(void)
{
    char s1[] = "Hello World";
    char* s2 = "Hello World";

    s1[0] = 'h';     // 가능
    // s2[0] = 'h';  // 불가

    printf("s1 : %s₩n", s1);

    s2 = "Hello World!!!";    // 가능
    // s1 = "Hello World!!!"; // 불가

    printf("s2 : %s₩n", s2);
```

s2의 경우 개별 요소를 수정할 수 없지만, 배열이 아니라 포인터 변수 이기 때문에, 새로운 배열의 포인터를 가르키면 된다.

포인터와 문자열

ROBIT
ROBOT SPORT GAME TEAM

- null 종료문자 ('\0')

C 언어에서 문자열은 null 종료 문자를 이용하여 끝을 표시한다.
null 종료 문자는 문자열의 끝을 나타내는 특별한 문자로, 문자열의 마지막에 추가된다.

포인터와 문자열

ROBIT
ROBOT SPORT GAME TEAM

- null 종료문자 (‘\0’)

```c
#include <stdio.h>

int main(void)
{
    char s1[] = "Hello World";
    char* s2 = "Hello World";

    s1[0] = 'h';     // 가능
    // s2[0] = 'h';  // 불가

    printf("s1 : %s₩n", s1);

    s2 = "Hello World!!!";    // 가능
    // s1 = "Hello World!!!"; // 불가

    printf("s2 : %s₩n", s2);

    for (int i = 0; s1[i] != '₩0'; i++)
    {
        printf("%c", s1[i]);
    }

    printf("₩n");

    for (int i = 0; i<sizeof(s1)/sizeof(s1[0]); i++)
    {
        printf("%c", s1[i]);
    }
```

포인터와 문자열

ROBIT
ROBOT SPORT GAME TEAM

- 문자열의 출력

```c
#include <stdio.h>

int main(void)
{
    char s1[] = "Hello World";
    char* s2 = "Hello World";

    s1[0] = 'h';     // 가능
    // s2[0] = 'h';  // 불가

    printf("s1 : %s₩n", s1);

    s2 = "Hello World!!!";    // 가능
    // s1 = "Hello World!!!"; // 불가

    printf("s2 : %s₩n", s2);

    for (int i = 0; s1[i] != '₩0'; i++)
    {
        printf("%c", s1[i]);
    }

    printf("₩n");

    for (int i = 0; i<sizeof(s1)/sizeof(s1[0]); i++)
    {
        printf("%c", s1[i]);
    }
```

%s 와 %c를 이용하여 문자/문자열을 출력할 수 있다.

- %s는 char* 즉, 주소값을 받아 '\n'이 나올 때까지 문자를 출력

- %c는 char 즉, 문자값 하나만 출력하기 때문에 개별적으로 for문을 활용하여 출력해 주어야 한다.

포인터와 문자열

- 문자열의 입력

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

void Input(char*);

int main(void)
{
    char s1[100];

    Input(s1);

    printf("%s", s1);

    return 0;
}

void Input(char* s)
{
    printf("문자열을 입력하시오 : ");
    scanf("%s", s);
}
```

왼쪽과 같이 scanf("%s", s); 을 통해 문자열을 입력받을 수 있다.

하지만 아래와 같이 공백을 만나기 전까지만 값을 s에 전달한다.

```
문자열을 입력하시오 : hello world
hello
```

포인터와 문자열

- 문자열의 입력

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

void Input2(char*);

int main(void)
{
    char s1[100];

    Input2(s1);

    printf("%s", s1);

    return 0;
}

void Input2(char* s)
{
    printf("문자열을 입력하시오 : ");
    scanf("%[^₩n]", s);
}
```

scanf("%[^₩n]", s); 를 사용하면,

아래와 같이 공백을 포함한 모든 문자를 입력받아 s에 전달한다.

```
문자열을 입력하시오 : hello world
hello world
```

포인터와 문자열

- 이중 포인터의 사용

```c
#include <stdio.h>

void SetStr(char** str);

int main(void)
{
    char* str;

    str = "I'am sad";

    SetStr(&str);

    printf("%s", str);

    return 0;
}

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

함수 외부에서 정의된 포인터값, 즉 주소값을 함수의 인수로 받아서 변경하고자 할때 사용. 즉, 포인터 변수가 가르키는 곳을 바꾸고자 함.

과제 공지

ROBIT
ROBOT SPORT GAME TEAM

1. 제출 기한 : 수요일 수업 전까지

2. 제출 형식 : 프로젝트 파일, 보고서(PDF)를 (로빛_18기_수습단원_이름)으로 압축 후
kwrobit2023@gmail.com 으로 제출
(보고서는 코드 설명과 실행 화면 첨부, 메일 제목은 C언어_n일차_이름)

3. 채점 기준:
1) 프로그램의 실행가능 여부
2) 교육하지 않은 C언어 개념 사용시 감점
3) 예외처리
4) 효율적인 코드 작성
5) 제출 형식

과제 1

ROBIT
ROBOT SPORT GAME TEAM

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.

과제 2

ROBIT
ROBOT SPORT GAME TEAM

비어있는 공집합 S가 주어졌을 때, 아래 연산을 수행하는 프로그램을 작성하시오.

- add x: S에 x를 추가한다. (1 ≤ x ≤ 20) S에 x가 이미 있는 경우에는 연산을 무시한다.
- remove x: S에서 x를 제거한다. (1 ≤ x ≤ 20) S에 x가 없는 경우에는 연산을 무시한다.
- check x: S에 x가 있으면 1을, 없으면 0을 출력한다.
- toggle x: S에 x가 있으면 x를 제거하고, 없으면 x를 추가한다. (1 ≤ x ≤ 20)
- all: S를 {1, 2, ..., 20} 으로 바꾼다.
- empty: S를 공집합으로 바꾼다.

```
연산을 선택하세요.
add : ~
remove : ~
check : ~
toggle : ~
all : ~
empty : ~

add3
input : add3

집합 : { 3, }

연산을 선택하세요.
add : ~
remove : ~
check : ~
toggle : ~
all : ~
empty : ~

add9
input : add9

집합 : { 3,  9, }

연산을 선택하세요.
add : ~
remove : ~
check : ~
toggle : ~
all : ~
empty : ~
```

조건 :
add,remove,check,toggle,all,empty 모두 포인터 함수로 변환

과제 2

ROBIT
ROBOT SPORT GAME TEAM

과제 2 의 조건

1. 동적 할당으로 최대 데이터 저장 크기 지정
2. 아래 조건 만족

- add x: S에 x(문자열)를 추가한다. (char형 데이터 저장) S에 x가 이미 있는 경우에는 연산을 무시한다.
- remove x: S에서 x(문자열)를 제거한다. S에 x가 없는 경우에는 연산을 무시한다.
- check x: S에 x(문자열)가 있으면 1을, 없으면 0을 출력한다.
- toggle x: S에 x(문자열)가 있으면 x를 제거하고, 없으면 x를 추가한다.
- all: S를 {1, 2, …. N} 으로 바꾼다.
- empty: S를 공집합으로 바꾼다.(메모리 해제 후 다시 최대 데이터 저장 크기 입력을 받아 반복)
- Size : S에 입력된 데이터의 개수 출력.

3. 최대 저장 가능한 배열 크기를 넘은 경우 예외 처리(isFull 함수 만들기)

조건 :
add,remove,check,toggle,all,empty 모두 포인터 함수로 변환

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
```

과제 4

두 문자열에 모두 포함된 가장 긴 공통 부분 문자열찾기.

조건
1. 두 입력은 char 형 데이터이며 동적 할당을 이용하여 크기를 지정해준다.
2. 동적 할당을 이용하여 저장 가능한 데이터의 길이보다 긴 문자열은 저장 가능한 만큼만 저장하고 나머지는 버린다.
3. 전체 시스템은 무한 루프로 구성 (quit을 입력하면 프로그램 종료)
4. 2 개의 배열과 결과를 저장할 배열, 즉 총 3 개의 Input 이 있으며, 반환형은 가장 긴 공통 부분의 문자열 길이를 담을 수 있는 int형인 함수를 사용한다.
5. 결과를 저장할 배열은 최초 2 개의 배열을 동적 할당 하는 과정에서 가장 짧은 길이를 가진 배열의 크기와 똑같은 크기로 동적 할당 한다.
6. 출력은 결과를 저장한 배열의 전체 원소와 함수의 반환으로 받은 문자열의 길이를 출력.

예시
입력1 : ajsdfkqWEefasdfegweoijo
입력2 : fkqWEefaEFAEFBVEWQGF

출력 : fkqWEefa / 8

과제 5

세 문자열에 모두 포함된 가장 긴 공통 부분 문자열찾기.

조건
1. 세 입력은 char 형 데이터이며 동적 할당을 이용하여 크기를 지정해준다.
2. 동적 할당을 이용하여 저장 가능한 데이터의 길이보다 긴 문자열은 저장 가능한 만큼만 저장하고 나머지는 버린다.
3. 전체 시스템은 무한 루프로 구성 (quit을 입력하면 프로그램 종료)
4. 세 개의 배열과 결과를 저장할 배열, 즉 총 4 개의 Input 이 있으며, 반환형은 가장 긴 공통 부분의 문자열 길이를 담을 수 있는 int형인 함수를 사용한다.
5. 결과를 저장할 배열은 최초 3 개의 배열을 동적 할당 하는 과정에서 가장 짧은 길이를 가진 배열의 크기와 똑같은 크기로 동적 할당 한다.
6. 출력은 결과를 저장한 배열의 전체 원소와 함수의 반환으로 받은 문자열의 길이를 출력.

예시
입력1 : ajsdfkqWEefasdfegweoijo
입력2 : fkqWEefaEFAEFBVEWQGF
입력3 : efjqeoijafkqWEeo

출력 : fkqWEe / 6