C교육 5주차

201702075 조수환

ARGOS

INDEX

ARGOS

C교육 5주차

4주차 과제 풀이

- 과제 1

ARGOS

과제 1

ARGOS

포인터와 역참조를 이용해서 str1의 문자들을 str2에 하나씩 넣는다.

```c
void copy(char* str1, char* str2) {
    while((*str2++ = *str1++) != '\0');
}

int main(void) {
    char str1[100] = "Copy This String";
    char str2[100];
    copy(str1, str2);
    printf("%s\n", str2);
    return 0;
}
```

```
sh2358@edu-argos:~/C-education/homework/week4$ ./homework1
Copy This String
```

C교육 5주차

동적할당

- 정의와 의의
- 가변길이 배열
- malloc
- 실습

ARGOS

정의와 의의

동적할당?

쉽게 풀어쓰자면 메모리 공간을 유동적으로 할당하는 것을 말한다.

예시를 들어보자. 동아리 행사에 참석하는 인원들의 학번을 입력받고 저장하는 프로그램이 필요하다. 얼마만큼의 메모리를 할당 해줘야 할까?

```
int student_code[100];
int attendee;
scanf("%d", &attendee);
for(int i=0 ; i<attendee ; i++) {
    scanf("%d", &student_code[i]);
}
```

몇 명이 참여 할지 모르기 때문에 최대 크기인 100(봄 톡방 인원)만큼 배열의 크기를 지정했다.

정의와 의의

하지만 아래의 예시처럼 참석자가 5명이라면

```
sh2358@edu-argos:~/C-education/practice$ ./malloc
5
202100000
202000000
201900000
201800000
201700000
```

배열 student_code[100]에 할당된 메모리 중 남은 95만큼의 공간을 쓰지 않는다.

바로 이러한 메모리 낭비를 줄이기 위해 동적할당을 사용한다.

그렇다면 참석자 수 처럼 입력받을 데이터 만큼의 공간만을 할애 할 수는 없을까?

가변길이 배열

입력받을 데이터 만큼의 공간을 할당하라고 한다면

아마 가장 먼저 아래와 같은 코드가 떠오를 것이다.

```c
int attendee;
scanf("%d", &attendee);
int student_code[attendee];
for(int i=0 ; i<attendee ; i++) {
    scanf("%d", &student_code[i]);
}
```

```c
for(int i=0 ; i<attendee ; i++) {
    printf("%d ", student_code[i]);
}
printf("\n");
```

위의 코드는 오류없이 정상적으로 돌아갈까?

오늘 배울 내용중에 malloc 이라는게 있었는데

자연스럽게 다음 설명으로 넘어가려면 아마 위의 코드는 안 돌아가지 않을까?

가변길이 배열

잘 돌아간다. 정확히는 1999년 개정된 C99부터 된다.

(그 전엔 배열의 크기를 변수로 지정하는 것이 불가능했다.)

```
sh2358@edu-argos:~/C-education/practice$ ./malloc
5
202100000
202000000
201900000
201800000
201700000
202100000 202000000 201900000 201800000 201700000
```

위처럼 실행 단계에서 배열의 크기를 정할 수 있는 배열을 가변길이 배열이라 한다.

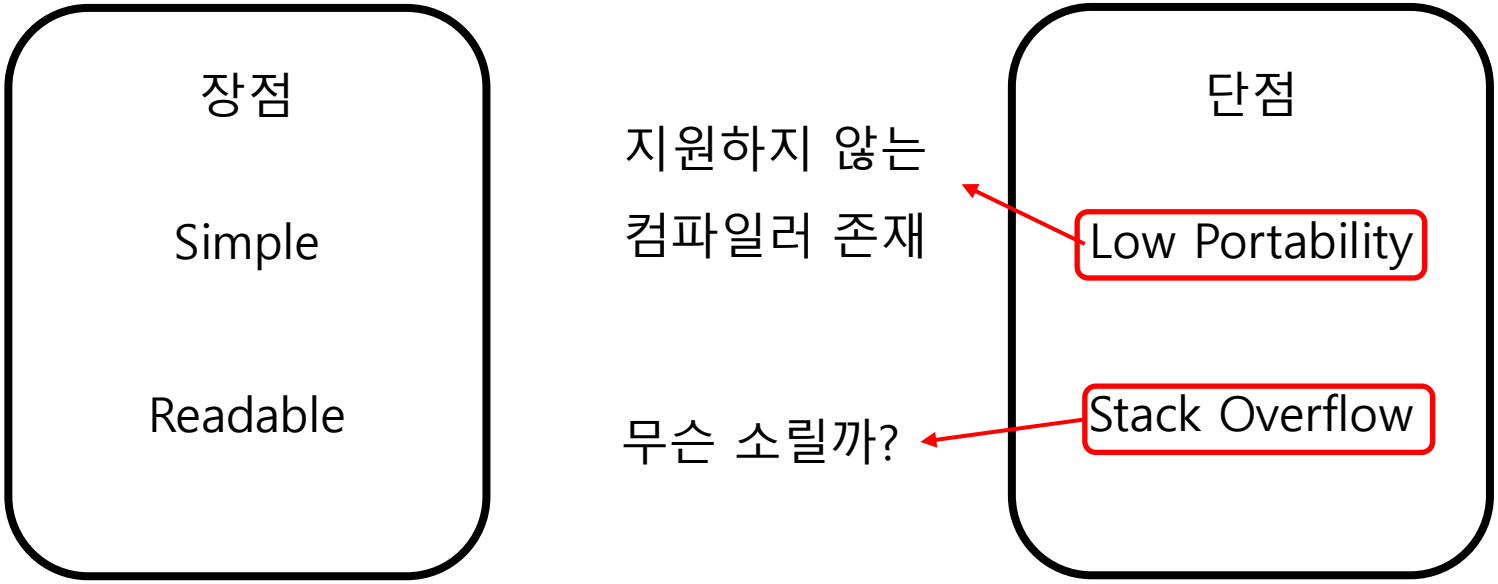
그럼 뒤에 나올 malloc은 무슨 쓸모가 있는걸까?

가변길이 배열

메모리 동적 할당이 필요한 모든경우에 가변길이 배열로 대처가 가능하다면,

오늘의 교육 내용은 여기서 끝일 것이다.

당연하게도, 각각의 방식에는 장단점이 있다.



가변길이 배열

Stack Overflow

아무튼 stack의 범위를 넘는 다는 것 같다.

stack은 자료구조 중 하나로 알고 있지만

여기서는 메모리 영역 중 하나를 의미한다.

```
int student_code[attendee];
```

가변 길이 배열은 스택 영역에 메모리 공간을 할당받는다.

가변길이 배열

아래의 코드를 컴파일하면서 스택에 메모리를 할당하는데..

```
int attendee;
scanf("%d", &attendee);
int student_code[attendee];
```

컴파일러는 student_code의 크기를 모른 채로 스택 메모리의 크기를 결정한다.

```
sh2358@edu-argos:~/C-education/practice$ gcc -o malloc malloc.c
sh2358@edu-argos:~/C-education/practice$
```

위처럼 아무런 에러메시지 없이 컴파일이 진행되었다.

그렇다고 문제가 없는것이 아니다. 만약 attendee에 충분히 큰 수가 입력된다면..

가변길이 배열

큰 수(1천만)를 입력하자 에러 메시지가 나오며 프로그램이 종료된다.

```
sh2358@edu-argos:~/C-education/practice$ ./malloc
10000000
Segmentation fault (core dumped)
```

여기서 segmentation fault는 건드리면 안되는 메모리 영역에 접근하거나

비 정상적인 방법으로 메모리에 접근을 시도 할 때 뜨는 에러다.

우린 그런 나쁜 짓을 한 기억이 없는데 어떻게 된걸까?

가변길이 배열

문제는 컴파일러가 배열의 크기를 모른 채로 메모리를 할당 한 것이다.

Stack Memory

대충 이만큼?

가변길이 배열의 크기는 컴파일 다음인 실행 단계에서 결정되므로

실행 단계에서 배열의 크기가 stack 메모리 크기를 넘어가면 프로그램이 죽는다.

Stack Memory student_code[10000000] ← (segfault)

정확한 금액을 모르고 예산을 짰다가 잔고가 박살난거랑 비슷하다.

malloc

ARGOS

malloc(memory allocation)

- 말 그대로 메모리를 할당하는 함수
- 메모리를 힙 영역에 할당함 (실행 단계에서 할당)
- 사용법 : [자료형]* [변수명] = (자료형*) malloc (크기)

```c
#include <stdio.h>
#include <stdlib.h>
```

malloc()이 선언된 헤더 파일

```c
int n;
scanf("%d", &n);
int* ptr = (int*) malloc(sizeof(int) * n);
```

int의 size(4byte) * n

malloc

```
sh2358@edu-argos:~/C-education/practice$ ./malloc
4
1
2
3
4
1 2 3 4
```

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int n;
    scanf("%d", &n);
    int* ptr = (int*) malloc(sizeof(int) * n);

    for(int i=0 ; i<n ; i++) {
        scanf("%d", ptr + i);
    }

    for(int i=0 ; i<n ; i++) {
        printf("%d ", *(ptr + i));
    }
    printf("\n");
    free(ptr)
    return 0;
}
```

free() ??

malloc으로 할당된 메모리 사용이 끝나면 free() 함수를 써서 해제!

malloc

malloc() 함수를 써 봅시다! malloc을 써서 n만큼의 공간을 할당받고,

입력받은 값의 제곱 값을 저장하라

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int n;
    scanf("%d", &n);
    /*
        write your code
                        */

    return 0;
}
```

```
sh2358@edu-argos:~/C-education/practice$ ./practice1
4
1
2
3
4
1 4 9 16
```

C교육 5주차

미세먼지 팁

- 표준 라이브러리
- Error type

ARGOS

표준 라이브러리

배열을 오름차순으로 정렬

문자열을 복사

실행 시간 측정

배열의 길이가 알고싶다

문자열 정수로 바꾸기

x의 제곱근을 구하라

무작위 숫자 받기

어디에서나 누구에게나 필요 할 법한 기능들

내가 꼭 일일이 구현해야해야 할까?

표준 라이브러리

ARGOS

이전 슬라이드의 예시들 뿐만 아니라 대부분 사람들이 필요로 하는

보편적 기능들은 표준 라이브러리에 구현 되어 있다.

<stdio.h>		**핵심 입력과 출력 함수들을 정의한다.**
<stdlib.h>		숫자 변환 함수들, 슈도 랜덤 숫자 생성 함수들, 메모리 할당, 프로세스 제어 함수들을 정의한다.
<stdnoreturn.h>	C11	반환하지 않는 함수들을 명시하기 위한
<string.h>		문자열 처리 함수들을 정의한다.
<tgmath.h>	C99	포괄형 수학 함수들을 정의한다.
<threads.h>	C11	다중 스레드들과 뮤텍스 그리고 제어 변수들을 관리하는 함수들을 정의한다.
<time.h>		데이터와 시간 처리 함수들을 정의한다.

우리가 쓰던 것들도 몇 개 보인다.

이 밖에도 많으니까 필요하면 검색해서 애용하도록 하자

Error type

프로그래밍을 하면서

내가 짠 코드의 양

VS

내가 본 버그의 양

전자가 더 많은 사람이 얼마나 있을까?

에러와 버그는 프로그래밍과 땔래야 땔수가 없다.

Error type

프로그래밍 4대 의문문

* 주관적인 생각입니다.

이게 왜 안되지?

이게 왜 되지?

아깐 잘 됐는데?

똑같은데 왜 난 안되지? (안 똑같음)

* 사실 궁금해서 물어보는게 아니다. 버그가 짜증나서 따지고 싶을 뿐

앞으로 코드 세스코가 될 여러분들을 위해 에러에 대한 내용을 준비했습니다.

Error type

프로그래밍 error type엔 크게 3가지가 있다.

- 구문 오류(Syntax Error)
- 실행 오류(Run-time Error)
- 논리 오류(Logical Error)

먼저 Syntax Error부터 보자

Error type

구문 오류(Syntax Error)

- 작성한 코드가 규정된 프로그래밍 문법에 맞지 않을 때에 발생
- 컴파일 단계에서 컴파일러가 어디가 왜 틀렸는지 짚어줍니다.
- 문법을 모른채로 쓰진 않으니 대부분 오타가 원인이다.

```c
#include <stdio.h>

int main(void) {
    int a = 4;
    int b = 3;
    printf("%d\n", a + b)
    return 0;
}
```

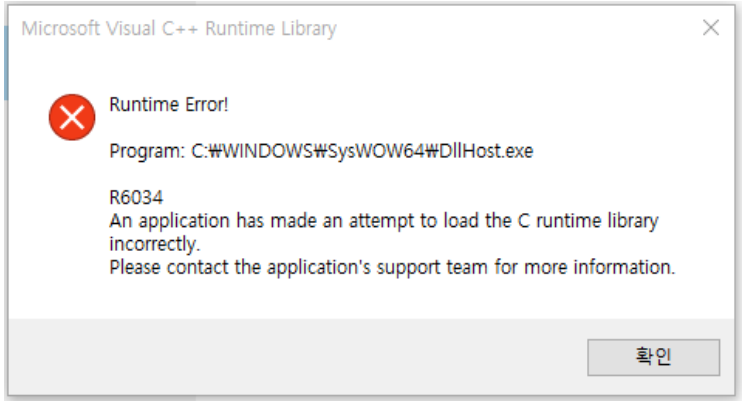
```
sh2358@edu-argos:~/C-education/practice$ gcc -o syntax syntax.c
syntax.c: In function ‘main’:
syntax.c:6:23: error: expected ‘;’ before ‘return’
    6 |  printf("%d\n", a + b)
      |                       ^
      |                       ;
    7 |  return 0;
      |  ~~~~~~
```

자세히 읽어보면 꽤 친절하다

Error type

실행 오류(Run-time Error)

- 말 그대로 프로그램 실행 시에 발생하는 오류
- 프로그램 작성 시의 설계 미숙이 주요 원인
- 드물게 기계적 결함으로도 발생



Error type

* Run-time Error 상습범들 *

- 무한 루프(조건이 잘못된 반복문, 탈출 없는 재귀 호출)
- 0으로 나누기
- Null Pointer (값이 필요한 부분에 Null(빈값)을 씀)
- Segmentation Fault (건드리면 안되는 메모리를 건드림)

이외에도 많은 원인이 있습니다. 의심스러운 곳부터 살펴줍시다.

보통 에러 문구와 원인을 출력해주니 구글 검색창에 복붙 하면

우리와 같은 Run-time Error 피해자들이 범인을 지목해 줄 겁니다.

Error type

ARGOS

논리 오류(Logical Error)

- 쉽게 말해서 오작동이다.
- 컴파일과 실행이 정상적으로 진행되나, 의도하지 않은 결과가 나옴
- 표시되는 에러메시지가 없어서 원인을 찾기가 매우 어렵다.

```c
#include <stdio.h>

int main(void) {
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);
    int avg = (a + b + c) / 3;
    if(avg <= a) printf("%d ", a);
    if(avg <= b) printf("%d ", b);
    if(avg <= c) printf("%d ", c);
    printf("\n");
    return 0;
}
```

```
sh2358@edu-argos:~/C-education/practice$ ./logical
3 4 6
4 6
```

3, 4, 6의 평균은 4.3333...이다.

Error type

논리 오류의 경우 컴파일과 실행 단계에서 문제가 없으므로 에러가 없다.

그래서 논리 오류를 찾을 때는 프로그램이랑 눈싸움을 하면 안된다.

의심 스러운 곳부터 하나 하나 건드리면서 수사를 해야 답을 찾을 수 있다.

```
int avg = (a + b + c) / 3;
printf("avg = %d\n", avg);
```

```
sh2358@edu-argos:~/C-education/practice$ ./logical
3 4 6
avg = 4
4 6
```

버그를 찾는게 힘든 만큼 해결하기만 하면

가장 많은 경험과 지식을 주는 버그다.

Q & A

Thank You for Listening