

고급 C프로그래밍 중간고사 대비자료

들어가는 말.

이번 중간고사는 매우 쉽게 내실 거라고 하셨다. 어려운 개념을 배우기도 했지만 개념적으로는 그렇게 까지 어려운 것은 없고 그냥 과제가 좀 어려운 것들이 많이 있었다.

이우신 교수님 스타일은 개념을 좀 집요하게 내시는 경향이 있기에 이 자료에서는 개념을 좀 많이 자세히 설명해 두었다.

정말 기본적인 개념만 넣어 놔으니 이 글을 읽고나서 교수님이 주신 자료 복기하면서 꼭 다시 공부해야 한다. 여기에 모든 내용이 있다고 생각하면 안됩니다.

1. 포인터

포인터의 정의는 변수의 주소를 담고 있는 변수이다. 변수를 만들게 되면 컴퓨터가 메모리에 자체적으로 공간을 만들어서 변수의 크기만큼 할당하게 된다.(int면 4byte, char면 1byte..)

과제에서 나오는 것 같이 아주 단순한 프로그램을 구현할 때는 굳이 포인터를 쓸 필요가 없다. 하지만 프로그램이 점점 커져서 막 구조체 선언해서 노드 여러 개 만들고 함수가 많아진다면 포인터를 쓰지 않고 모든 변수들을 매개변수 선언하기에는 불가능에 가까울 정도로 복잡 해진다.

그래서 포인터를 사용한다. 또 동적할당을 사용하여 필요한 만큼의 메모리만 딱 사용할 수도 있어서 메모리가 아주 부족한 임베디드 환경에서는 매우 중요하다.(C언어는 보통 임베디드 환경에서 사용한다.)

포인터는 왜 사용하는지를 생각해보면 매우 쉽게 이해할 수 있다. 포인터 변수에는 변수의 주소가 담겨있기 때문에 포인터 변수를 사용하여 변수의 주소에 접근을 한 후 그 값을 건드릴 수 있다. 이것 역으로 참조한다고 하여 역참조라고 부른다.

<예제>

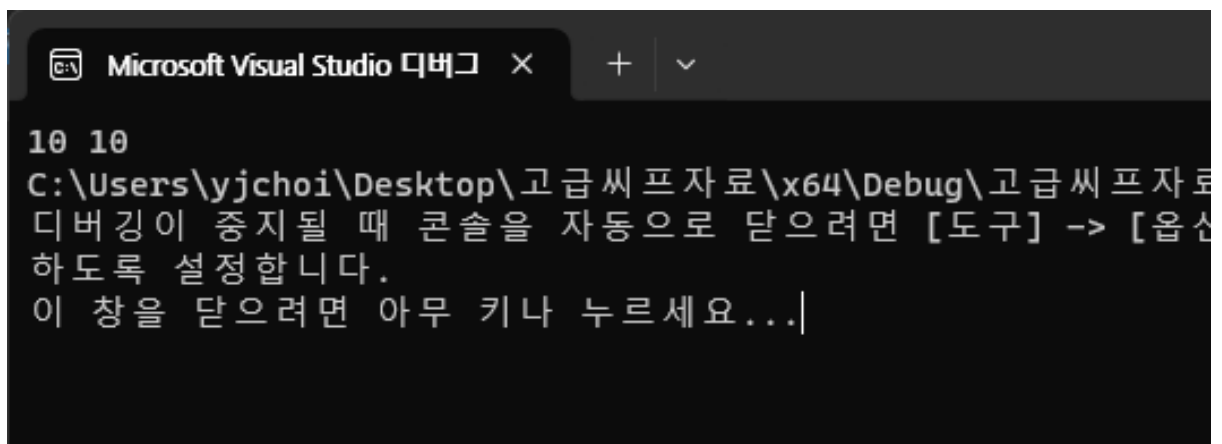
```
#include <stdio.h>

int main() {
    int a = 10; //사용할 변수에 10을 넣어서 선언
    int* p = &a; //p라는 포인터 변수에 a의 주소값 저장(&는 주소연산자)

    printf("%d %d", a, *p); //a==*p *p의 의미는 p라는 변수에 있는 값인
    //a의 주소에 접근해서 역참조 했다는 의미이다

    return 0;
}
```

<실행결과>



The screenshot shows the Microsoft Visual Studio Debug Console. The title bar reads "Microsoft Visual Studio 디버그" with a close button. The console output is as follows:

```
10 10
C:\Users\yjchoi\Desktop\고급씨프자료\x64\Debug\고급씨프자료
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션]
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...|
```

2. 동적 할당

앞서 포인터를 사용하는 이유 중에 메모리를 필요한 만큼만 할당해서 사용할 수 있다고 말했다. 포인터 변수에 메모리를 할당해서 사용하는 것을 동적할당이라고 한다.

'stdlib.h' standard library header file에 malloc()이라는 memory allocation 함수가 있는데 이를 이용하면 포인터 변수에 메모리를 할당해서 사용할 수 있다.

포인터 변수에 메모리를 할당해서 사용한다는 게 어떤 의미일까? 지금까지는 변수를 선언하고 포인터 변수를 선언해서 변수의 주소를 포인터 변수에 담고 그 후에 역참조해서 사용했다.

즉 1개의 변수를 사용하기 위해 실질적으로 그냥 변수와 포인터 변수 2개를 선언해야 했었다. 하지만 동적할당을 사용하면 그냥 변수를 선언하지 않고 포인터 변수에 메모리를 넣어줘서 그냥 변수처럼 역참조만 해서 사용할 수 있다.

포인터 변수도 int형이면 4바이트 할당된다고 했는데 동적할당을 왜 또 해야 하는지 의문을 가질 수 있다. 포인터 변수를 선언하면서 할당되는 4바이트는 주소를 담는 데에 사용하는 메모리고 정수형 값을 사용하는 메모리가 아니다.

그래서 지금까지 정수형 값을 사용하는 메모리를 강제로 만들어주고 그 주소를 포인터 변수에 넣어서 강제로 엮어서 사용했던 것이다.

동적할당을 하면 이 과정을 생략할 수 있다. 즉 값을 담은 메모리는 직접 포인터 변수에 할당해주는 과정이 동적할당이다.

조금 더 심화로 들어가면 그냥 int a;나 int *p;등을 선언할 때 메모리는 메모리 중 스택(stack)이라는 곳에 할당된다. 하지만 동적할당으로 생성시킨 메모리는 힙(heap)이라는 부분에 할당되는데 스택(stack)은 자동으로 메모리 해제를 해주지만 힙(heap)부분은 할당을 직접 해줘야 한다.

따라서 동적할당을 하여 메모리를 할당했으면 free()함수를 사용하여 반드시 메모리를 해제해야 한다. 안 하면 메모리 누수가 나서 코드의 문제가 생긴다.

동적할당을 하는 방법은 *포인터 = malloc(사용할 메모리크기); 이런 식으로 할당한다. 보통은 포인터 변수를 선언하면서 바로 할당해주지만 포인터 변수를 선언하고도 할 수 있다.

보통 malloc()함수를 쓰지만 메모리를 할당하고 0으로 초기화할 때는 calloc()함수를 사용하기도 한다.

이미 할당을 한 상황에서 코드를 짜다가 메모리가 부족할 때 동적할당한 부분을 찾기 어려울 정도로 프로젝트가 크다면 realloc()을 사용해서 필요한 만큼 더 할당할 수도 있다.

포인터 = (포인터 변수의 데이터형)realloc(포인터 변수, 추가할 메모리크기);

이런 식으로 다시 할당한다.

<예제>

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    //기존에 포인터 변수에 값을 넣어서 사용하는 방법
    int a; //a라는 4byte짜리 변수 선언
    int* p = &a; //p라는 포인터 변수에 a의 주소값 저장 -> 4byte메모리 사용 가능해짐
    *p = 10;

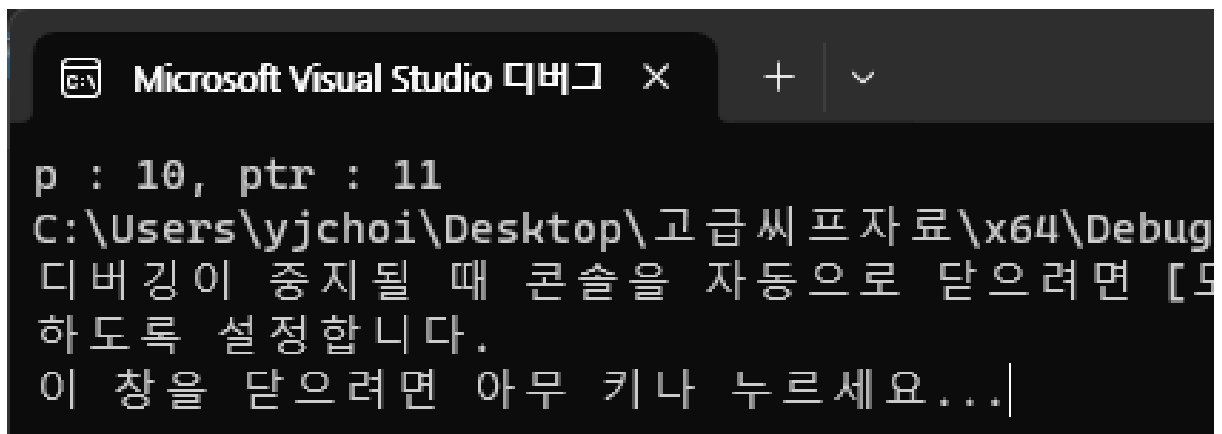
    //동적할당
    int* ptr = malloc(sizeof(int)); //ptr이라는 포인터 변수에 int만큼 즉 4byte 메모리 할당
    *ptr = 11;

    printf("p : %d, ptr : %d", *p, *ptr);

    //heap부분 건드렸으니깐 즉 동적할당했으니깐 직접 메모리해제
    free(ptr);

    return 0;
}
```

<출력 결과>



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio 디버그" with a close button. The console output displays the values of the pointers: "p : 10, ptr : 11". Below this, there is a system message in Korean: "C:\Users\yjchoi\Desktop\고급씨프자료\x64\Debug 디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도움말] 메뉴를 참조하십시오." followed by "이 창을 닫으려면 아무 키나 누르세요..." with a cursor.

```
C:\Users\yjchoi\Desktop\고급씨프자료\x64\Debug
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도움말]
메뉴를 참조하십시오.
이 창을 닫으려면 아무 키나 누르세요...
```

3. 표준 입출력

지금까지 쓰던데 표준 입출력이다. `stdio.h`을 습관처럼 선언해서 사용했다. 이 헤더파일이 무슨 헤더파일이라면 `standard input output`의 관련된 헤더파일이다. 즉 이 헤더파일로 우리는 표준 입출력을 해왔다.

`printf`, `scanf`등 터미널에 띄우는 모든 함수가 표준 입출력 함수이다. 이상한 함수들을 몇 개 수업 때 소개하셨는데 평생 써 본적 없지만 그냥 수업 때 했으니까 정리만 해 두었다.

입력 함수	설명
<code>getch()</code>	키보드를 통해 문자 하나를 입력받으며, 입력한 내용을 모니터에 보여주지 않는다.
<code>getche()</code>	키보드를 통해 문자 하나를 입력받으며, 입력한 내용을 모니터에 보여준다.
<code>getchar()</code>	사용자가 키보드로 <code>Enter</code> 를 누를 때까지 입력한 것을 메모리(버퍼)에 모두 저장해놓고(<code>Enter</code> 도 저장됨) 그중에서 한 문자만 꺼낸다.
<code>putchar(문자형 변수)</code>	표준 출력 장치(모니터)에 문자 하나를 출력한다.
<code>putch(문자형 변수)</code>	<code>putchar()</code> 와 기능이 동일하다.

4. 파일 입출력

말 그대로 터미널에서 하던 작업을 파일로 하는 것이다. txt파일이나 hwp 등 확장자를 사용하여 사용하고 싶은 종류의 파일로 작업할 수 있다.

파일 포인터 변수를 선언하고 파일을 모드를 선택하여 열어주고 파일에 어떤 작업을 한 후 파일을 닫아주면 된다.

모드	설명
"r"	읽기 모드로 파일을 연다. 만약 파일이 존재하지 않으면 오류가 발생한다.
"w"	쓰기 모드로 새로운 파일을 생성한다. 파일이 이미 존재하면 기존의 내용이 지워진다.
"a"	추가 모드로 파일을 연다. 만약 기존의 파일이 있으면 데이터가 파일의 끝에 추가된다. 파일이 없으면 새로운 파일을 만든다.
"r+"	읽기 모드로 파일을 연다. 쓰기 모드로 전환할 수 있다. 파일이 반드시 존재하여야 한다.
"w+"	쓰기 모드로 새로운 파일을 생성한다. 읽기 모드로 전환할 수 있다. 파일이 이미 존재하면 기존의 내용이 지워진다.
"a+"	추가 모드로 파일을 연다. 읽기 모드로 전환할 수 있다. 데이터를 추가하면 EOF 마커를 추가된 데이터의 뒤로 이동한다. 파일이 없으면 새로운 파일을 만든다.
"t"	텍스트 파일 모드로 파일을 연다.
"b"	이진 파일 모드로 파일을 연다.

➔ 이 표 보고 사용하고 싶은 모드를 fopen()할 때 선언 해주면 된다.

<예시>

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

int main() {
    //fp라는 파일 포인터 변수 선언
    FILE* fp;

    //파일 오픈함수를 사용해서 test.txt파일을 만들면서 읽기+쓰기 모드로 열어준다
    fp = fopen("test.txt", "w+");

    //fprintf함수를 사용하여 fp에 들어가있는 test.txt에 hello입력
    fprintf(fp, "hello");

    //작업 끝났으니깐 닫아주기
    fclose(fp);

    return 0;
}
```

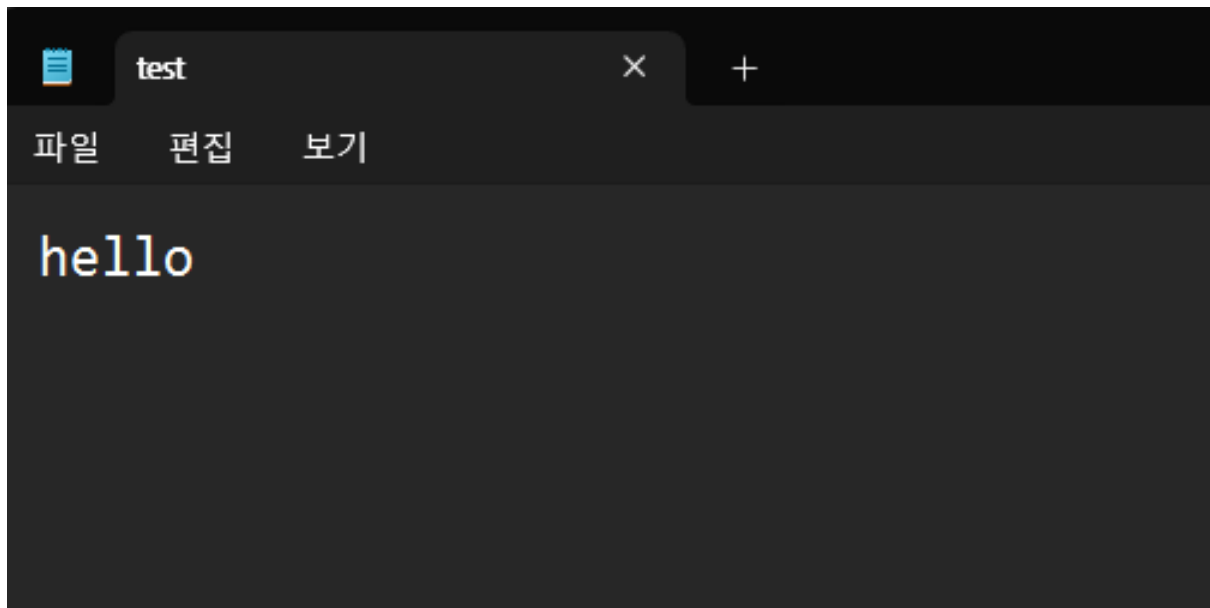
->코드



이름	수정한 날짜	유형	크기
.vs	2023-10-23 오전 1:25	파일 폴더	
x64	2023-10-23 오전 1:30	파일 폴더	
glaemfjddy	2023-10-23 오전 2:20	C Source File	1KB
test	2023-10-23 오전 2:20	텍스트 문서	1KB
고급씨프자료.sln	2023-10-23 오전 1:25	Visual Studio Sol...	2KB
고급씨프자료.vcxproj	2023-10-23 오전 1:30	VCXPROJ 파일	7KB
고급씨프자료.vcxproj.filters	2023-10-23 오전 1:30	VC++ Project Filt...	1KB
고급씨프자료.vcxproj.user	2023-10-23 오전 1:25	Per-User Project ...	1KB

->수업때 보니깐 막 파일 만들어서 저장하고 열어서 쓰는 사람 있던데 그냥 fopen함수 쓰면 자동으로 만들어주니깐 그럴 필요 없음. 만들어주는 위치도 막 쓰는 사람들 많던데 본인 프로젝

트 안에 다 들어있음. c파일 있는 프로젝트 폴더에 자동으로 생성됨.



->파일 결과