C언어 포트폴리오

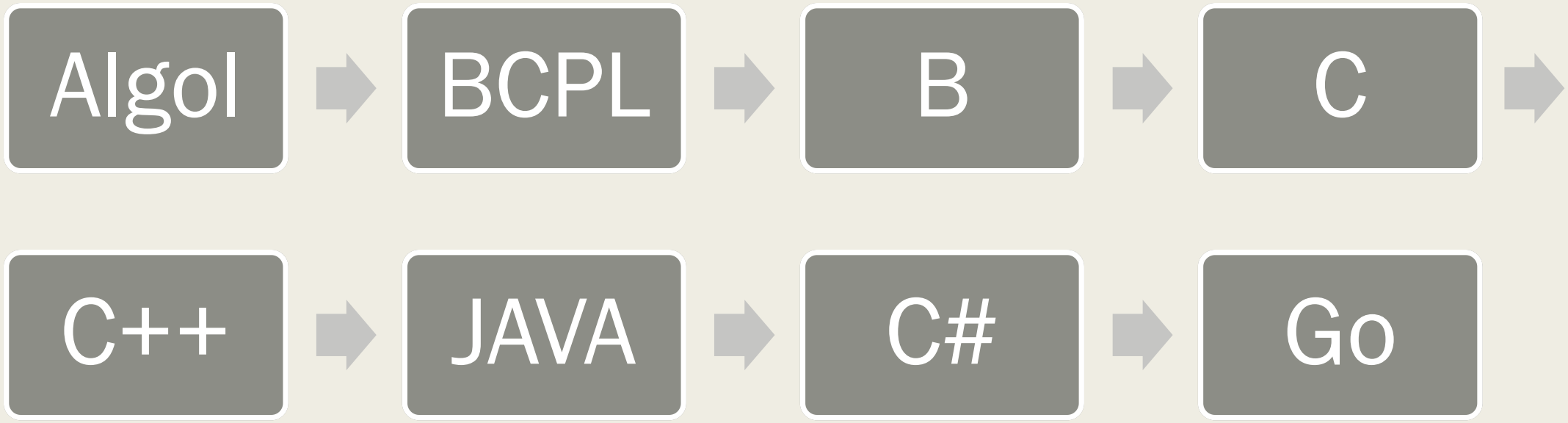
20192623 이준모

목차

C언어란?

C언어는 1972년 데니스 리치(Dennis Ritchie)가 개발한 프로그래밍 언어이다. 데니스 리치는 유닉스 운영체제를 개발을 계획하고 있었으며 이를 위해 어셈블리 언어 정도의 속도를 내며, 좀 더 쉽고, 서로 다른 CPU에서도 작동되는 프로그래밍 언어가 필요했다. 그래서 개발된 언어가 C언어이다. C언어는 B언어에서 유래된 프로그래밍 언어이다.

C언어 이후의 프로그래밍 발전 단계



C언어의 특징

- 절차 지향 언어
  - C언어는 함수 중심으로 구현되는 절차지향언어이다. 절차지향 언어는 시간의 흐름에 따라 정해진 절차를 실행한다는 의미로 C언어는 문제의 해결 순서와 절차의 표현과 해결이 쉽도록 설계된 프로그램 언어다.
- 간결하고 효율적인 언어
  - C언어는 간결하고 효율적인 언어이다. 함수의 재귀호출이 가능하고 시스템의 세세한 부분까지 제어할 수 있으며 포인터와 메모리 관리 기능 또한 갖고 있다. C언어로 작성된 프로그램은 크기도 작으며 메모리도 적게 효율적으로 사용하여 실행 속도가 빠르다는 장점이 있다.
- 이식성이 좋은 프로그램 언어
  - C언어는 다양한 CPU와 개발플랫폼의 컴파일러를 지원함으로 이식성이 좋다
- 다소 어렵다는 단점
  - C언어는 어려운 개념들이 있어 다소 배우기 어렵다는 단점이 있다.

함수의 기본 정보

```c
#include <stdio.h>

int main(void)                  //함수의 머리로 int는 결과값의 유형, void는 함수의 입력이 없음을 의미한다
{                               // 함수의 시작을 알리는 괄호
    printf("Hello World!₩n");   //함수 호출

    return 0;                   //함수 반환값이 0이라는 것을 뜻함
                                //괄호의 내부가 함수의 몸체이다
}                               // 함수의 끝을 알리는 괄호
```

- 함수는 함수의 머리와 함수의 몸체로 구성되어 있다.
  - 함수의 머리는 결과값의 유형, 함수이름, 매개변수인 입력변수 나열을 각각 표시한다.
  - 함수의 몸체는 함수의 머리 이후 {}안에 구현된 부분을 의미한다.

11단원
문자와 문자열

- 문자는 한 글자를 작은 따옴표로 나타내고 이것을 문자 상수라고 한다.
- 문자열은 문자들이 모인 것을 의미하며 큰 따옴표로 나타낸다. 문자열을 작은 따옴표로 나타내려고 하면 오류가 발생한다.
- 문자는 char에 문자열은 문자배열을 이용하여 저장한다. 문자열은 저장할 때 마지막에 \0을 저장해 주어야 하기 때문에 크기가 1이 커진다.

문자와 문자열 코드

코드

```c
#include <stdio.h>

int main(void)
{
    char ch = 'A';                          //문자는 char에 저장하고 작은 따옴표를 사용한다
    printf("%c %d₩n", ch, ch);

    char java[] = { 'J', 'A', 'V', 'A', '₩0' };//문자열을 선언할 때 마지막에 ₩0을 저장한다
    printf("%s₩n", java);

    char c[] = "C language";                //문자열은 큰따옴표로 저장 할 수도 있다
    printf("%s₩n", c);                      //이 경우에는 마지막에₩0이 자동으로 저장된다

    char csharp[5] = "C#";                  //크기를 미리 정하고 저장할 수도 있다
    printf("%s₩n", csharp);                 //남은 부분은 ₩0으로 채워진다.

    printf("%c%c₩n", csharp[0], csharp[1]);

    system("pause");
    return 0;
}
```

실행 결과

```
A 65
JAVA
C language
C#
C#
계속하려면 아무 키나 누르십시오 . . .
```

문자 입력 함수

- getchar() - 라인 버퍼링을 사용하므로 엔터키를 눌러야 반응을 한다.

```
printf("₩n문자를 누를 때마다 두 번 출력 >>₩n");
while ((ch = _getche()) != 'q')
    putchar(ch);
```

```
문자를 계속 입력하고 Enter를 누르면     >>
hello
hello
```

- getche() – 헤더파일 conio.h가 필요하며 버퍼를 사용하지 않고 문자를 바로 입력 할 수 있다.

```
printf("문자를 계속 입력하고 Enter를 누르면 >>₩n");
while ((ch = getchar()) != 'q')
    putchar(ch);
```

```
문자를 누를 때마다 두 번 출력   >>
hheellllloo
```

- getch() – getche와 비슷하지만 입력한 문자가 화면에 보이지 않는다.

```
printf("₩n문자를 누르면 한 번 출력  >>₩n");
while ((ch = _getch()) != 'q')
    _putch(ch);
printf("₩n");
```

```
문자를 누르면 한 번 출력        >>
helo
```

문자열 입출력 함수

- gets() – 문자열을 버퍼에 저장하며 첫 문자의 주소값을 반환해준다. 엔터키를 입력하기 전까지의 모든 문자열을 입력 받으며 엔터값은 \0으로 대체하여 저장한다.

```
while (gets(line))
    puts(line);
printf("₩n");
```

```
gets
gets
```

- gets_s() – gets와 동일하지만 두 번째 인자에 버퍼의 크기를 지정해 주어야한다.

```
while (gets_s(line, 101))
    puts(line);
printf("₩n");
```

```
gets_s
gets_s
```

- 위의 결과 값에서 출력은 puts로 한 것으로 puts는 문자열에 저장된 마지막 \0 값을 \n값으로 반환하여 출력한다.

gets와 puts는 처리속도가 빠르다는 장점이 있다.

문자열 관련 함수들

```
char src[50] = "https://www.visualstudio.com";
char dst[50];

printf("문자배열 src = %s₩n", src);
printf("문자열크기 strlen(src) = %d₩n", strlen(src));
memcpy(dst, src, strlen(src) + 1);
printf("문자배열 dst = %s₩n", dst);
```

```
문자배열 src = https://www.visualstudio.com
문자열크기 strlen(src) = 28
```

memcpy()

dst 값에 src에서 strlen(src)+1 만큼의 문자열을 저장한다

```
s1 = "java";
s2 = "jav";
printf("strcmp(%s,%s) = %d₩n", s1, s2, strcmp(s1, s2));
```

```
strcmp(java,jav) = 1
```

strcmp()

두 문자열을 비교하여 같으면 0 앞이 크면 양수 뒤가 크면 음수를 반환한다

```
char *s1 = "java";
char *s2 = "java";
printf("strncmp(%s,%s,%d) = %d₩n", s1, s2, 3, strncmp(s1, s2, 3));
```

```
strncmp(jav,java,3) = 0
```

strncmp()

strcmp와 동일하나 매개변수로 받은 상수값의 크기만큼만 비교한다.

문자열 복사와 연결

- strcpy()

```c
int main(void)
{
    char dest[80] = "Java";
    char source[80] = "C is a language.";

    printf("%s\n", strcpy(dest, source));
    printf("%s\n", strncpy(dest, "C#", 2));

    printf("%s\n", strncpy(dest, "C#", 3));

    system("pause");
    return 0;
}
```

```
C is a language.
C#is a language.
C#
```

strcpy와 strncpy는 앞 인자에 뒷 인자의 문자열을 복사하여 저장하는 함수이다. strncpy는 몇 개의 문자를 저장할 것인지 정할 수 있다.

- strcat()

```c
int main(void)
{
    char dest[80] = "C";

    printf("%s\n", strcat(dest, " is "));
    printf("%s\n", strncat(dest, "a java", 2));
    printf("%s\n", strcat(dest, "procedural "));
    printf("%s\n", strcat(dest, "language."));

    system("pause");
    return 0;
}
```

```
C is
C is a
C is a procedural
C is a procedural language.
```

strcat과 strncat은 문자를 연결해주는 함수로 문자열을 연결하여 앞 문자열의 주소를 반환해준다. strncat은 뒷 문자열을 어디까지 연결할 것인지 정할 수 있다.

문자열 분리와 검색

- strtok()

```
int main(void)
{
    char str1[] = "C and C++₩t language are best!";
    char *delimiter = " ,₩t!";

    printf("문자열 ₩"%s₩"을 >>₩n", str1);
    printf("구분자[%s]를 이용하여 토큰을 추출    >>₩n", delimiter);
    char *ptoken = strtok(str1, delimiter);
    while (ptoken)
    {
        printf("%s₩n", ptoken);
        ptoken = strtok(NULL, delimiter);
    }

    system("pause");
    return 0;
}
```

```
문자열 "C and C++        language are best!"을  >>
구분자[ ,       !]를 이용하여 토큰을 추출      >>
C
and
C++
language
are
best
계속하려면 아무 키나 누르십시오 . . .
```

strtok는 앞 문자열을 뒷 문자열을 구분자로 하여 토큰을 만들어 반환해주는 함수이다.

- strlen(),strlwr()/strupr(),strstr()/strchr()

```
int main(void)
{
    char str[] = "JAVA 2017 go c#";
    printf("%d₩n", strlen("java"));
    printf("%s, ", _strlwr(str));
    printf("%s₩n", _strupr(str));

    printf("%s, ", strstr(str, "VA"));
    printf("%s₩n", strchr(str, 'A'));

    system("pause");
    return 0;
}
```

```
4
java 2017 go c#, JAVA 2017 GO C#
VA 2017 GO C#, AVA 2017 GO C#
```

strlen는 문자열의 길이를 반환해주고 strlwr, strupr은 각각 문자열을 소문자 대문자로 변환해준다. strstr, strchr은 각각 문자열, 문자가 처음 나타나는 곳의 주소를 반환해준다.

여러 개의 문자열

- 문자 포인터 배열

char *pa[] = {“JAVA”, “C#”, “C++”};

이와 같은 방식으로 배열로 여러 문자열을 저장할 수 있으며 pa[0][0] = ‘J’ pa[1][1] = ‘#’ 과 같은 방식으로 저장된다.

- 이차원 문자 배열

char ca[][5] = {“JAVA”, “C#”, “C++”};

이차원 문자배열도 포인터 배열과 비슷하게 사용되지만 열 크기를 지정해 주어야 한다. 열 크기보다 저장되는 값이 작을 경우 빈 공간은 \0이 저장된다.

명령행 인자

- 명령행에서 입력하는 문자열을 프로그램으로 전달하는 방법이 명령행 인자를 사용하는 방법이다.
- 명령행 인자를 사용하기 위해서는 int main(int argc, char *argv[])라고 기술해주어야 한다. argc는 명령행에서 입력한 문자열의 수이며 argv[]는 명령행에서 입력한 문자열을 전달 받는 문자 포인터 배열이다.

```
for (i = 0;i < argc;i++)
        printf("argv[%d] = %s\n", i, argv[i]);
```

명령 인수를 C#, C++, JAVA로 설정하고 위 구문을 실행 시켰다면 자기 자신인 실행프로그램의 이름과 위치값과 명령 인수 값들이 결과에 표시 될 것이다.

12단원
변수 유효범위

- 변수를 참조할 수 있는 유효한 범위를 변수의 유효범위라고 하며 유효범위는 지역 유효범위와 전역 유효 범위로 나뉜다.
- 지역 유효 범위는 함수나 블록 내부에서 선언되어 그 지역에서 변수의 참조가 가능한 범위이다.
- 전역 유효 범위는 파일에서만 참조할 수 있는 범위와 프로젝트 내부에서 참조할 수 있는 범위가 있다.

지역변수

- 지역변수는 선언 문장 이후에 함수에 사용할 수 있다.
- 함수의 매개변수 또한 지역변수이다.
- 지역변수는 선언 후 초기화를 해주지 않으면 쓰레기값이 저장된다.
- 지역변수는 stack에 할당되며 사용되는 함수가 종료되면 stack에서 제거된다.

```
for (int m = 0, sum = 0; m < 3; m++)
{
    sum += m;
    printf("₩t%d %d₩n", m, sum);
}
```

만약 for문이 다음과 같이 있다면 m, sum은 for문의 지역변수 이므로 for문 내부 이외의 다른 곳에서 참조 할 수 없다.

```
void sub(int param)
{
    auto int local = 100;
    printf("₩t%d %d₩n", param, local);
}
```

함수의 매개변수인 param과 내부의 local 또한 지역변수 이므로 sub 함수 내부에서만 사용할 수 있다.

전역변수

- 전역변수는 함수 외분에서 선언되는 변수로 외부변수라고도 부르며 프로젝트의 모든 함수나 블록에서 참조할 수 있다.
- 전역변수는 선언될 때 자동으로 초기값을 0으로 설정한다.
- 만약 함수 내부에서 전역변수와 같은 이름의 지역변수를 선언했으면 그 함수 내부에서 참조할 때 함수 내부의 변수값을 가져온다.
- extern을 이용하여 프로젝트의 다른 파일에서도 참조 가능하다.

```c
#include <stdio.h>

double getArea(double);
double getCircum(double);

double PI = 3.14;
int gi;

int main(void)
{
    double r = 5.87;
    const double PI = 3.141592;

    printf("면적: %.2f₩n", getArea(r));
    printf("둘레1: %.2f₩n", 2 * PI*r);
    printf("둘레2: %.2f₩n", getCircum(r));
    printf("PI: %f₩n", PI);
    printf("gi: %d₩n", gi);

    system("pause");
    return 0;
}

double getArea(double r)
{
    return r * r*PI;
}
```

```c
extern double PI;

double getCircum(double r)
{
    return 2 * r*PI;
}
```

프로젝트의 다른 변수에서 extern을 사용하여 전역변수 pi 사용

```
면적: 108.19
둘레1: 36.88
둘레2: 36.86
PI: 3.141592
gi: 0
```

함수 밖에서 전역함수 선언 gi는 초기값 0

함수 내부에서도 pi를 선언 했으므로 내부의 pi 값을 가져온다.

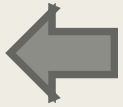
기억부류

- 변수는 기억부류가 있고 이에 따라 메모리영역, 할당과 제거 시기가 결정된다.
- 기억부류의 종류는 auto, register, static, extern으로 총 4가지이다.

auto	지역변수에 이용 일반 지역변수로 생략될 수 있다	static	지역변수와 전역변수 모두에 이용 정적변수로 생성된 이후에는 메모리에서 제거되지 않는다 초기화는 한번만 실행되고 초기값을 지정하지 않으면 0이 저장된다
register	지역변수에 이용 레지스터 변수는 cpu 내부의 레지스터에 할당되므로 처리 속도가 빠르다 주소 연산자 &를 사용할 수 없다	extern	전역변수에 이용 extern은 전역변수를 사용할 때 선언하다

register 변수

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

int main(void)
{
    register int sum = 0;

    int max;
    printf("양의 정수 입력  >>  ");
    scanf("%d", &max);

    for (register int count = 1; count <= max; count++)
        sum += count;

    printf("합 : %d₩n", sum);

    system("pause");
    return 0;
}
```



sum은 레지스터 변수이므로 cpu레지스터에 저장된다

```
양의 정수 입력  >>      100
합 : 5050
```

그러므로 처리할 것이 많을 때도 금방 처리할 수 있다

정적 지역변수

```c
#include <stdio.h>

void increment(void);

int main(void)
{
    for (int count = 0; count < 3; count++)
        increment();
    system("pause");
}

void increment(void)
{
    static int sindex = 1;
    auto int aindex = 1;

    printf("정적 지역변수 sindex : %2d,\t", sindex++);
    printf("자동 지역변수 aindex : %2d\n", aindex++);
}
```

```
정적 지역변수 sindex :  1,      자동 지역변수 aindex :  1
정적 지역변수 sindex :  2,      자동 지역변수 aindex :  1
정적 지역변수 sindex :  3,      자동 지역변수 aindex :  1
계속하려면 아무 키나 누르십시오 . . .
```

sindex는 정적 지역변수이므로 aindex와 달리 함수가 계속 호출되더라도 메모리에 남아있으므로 값이 증가된다

정적 전역변수

```c
#include <stdio.h>

static int svar;
int gvar;

void increment();
void testglobal();

int main(void)
{
    for (int count = 1; count <= 5; count++)
        increment();
    printf("함수 increment()가 총 %d번 호출되었습니다.\n", svar);

    testglobal();
    printf("전역 변수 : %d\n", gvar);

    system("pause");
}

void increment()
{
    svar++;
}
```

```c
void testglobal()
{
    extern gvar;
    gvar = 10;
}
```

```
함수 increment()가 총 5번 호출되었습니다.
전역 변수 : 10
계속하려면 아무 키나 누르십시오 . . .
```

svar은 정적 전역변수이므로 정적 지역변수와 마차가지로 값이 증가되었다. testglobal에서 extern을 이용하여 전역변수를 사용하였지만 정적 전역변수는 다른 파일에서는 사용할 수 없다.

메모리 영역

- 메인 메모리 영역으로는 data, heap, stack 세가지가 있다.
  - data 영역은 전역변수와 정적변수가 할당된다.
  - heap 영역은 동적 할당되는 변수가 할당된다.
  - stack 영역은 함수 호출에 의한 형식 매개변수, 함수 내부의 지역변수가 할당된다.
- 메모리 영역은 유효범위와 생존기간을 결정한다.
- 데이터 영역은 주소가 낮은 값에서 높은 값으로 할당되며 프로그램이 시작될 때 고정된 메모리 값이 정해진다.
- 스택 영역은 주소가 높은 값에서 낮은 값으로 저장된다.
- 힙 영역은 낮은 값에서 높은 값으로 사용하지 않는 공간이 동적으로 할당된다.

변수의 종류

<table>
<tr><th>선언위치</th><th>상세종류</th><th>키워드</th><th>유효범위</th><th>기억장소</th><th>생존시간</th></tr>
<tr><td rowspan="2">전역</td><td>전역변수</td><td>extern</td><td>프로그램 전역</td><td rowspan="2">메모리(데이터)</td><td rowspan="2">프로그램
실행 시간</td></tr>
<tr><td>정적 전역 변수</td><td>static</td><td>파일 내부</td></tr>
<tr><td rowspan="3">지역</td><td>정적 지역변수</td><td>static</td><td rowspan="3">함수나 블록
내부</td><td></td><td></td></tr>
<tr><td>레지스터 변수</td><td>register</td><td>레지스터</td><td rowspan="2">함수 또는 블록
실행 시간</td></tr>
<tr><td>자동 지역변수</td><td>auto</td><td>메모리(스택)</td></tr>
</table>

전역변수는 동일 파일 다른 파일의 외부 함수에서 사용 가능하고 정적 전역 변수는 동일 파일 다른 함수에서 사용 가능하다 나머지 변수들을 외부에서 사용할 수 없다.

전역변수, 정적 변수들은 프로그램 시작 시 자동으로 초기값 0이 저장되고 나머지는 실행될 때마다 쓰레기값이 저장된다.

전역변수, 정적 변수들은 프로그램 종료 시 메모리에서 제거되고 나머지는 각 함수가 종료될 때 메모리에서 제거된다.

13단원
구조체와 공용체

- 구조체란 정수나 문자, 실수나 포인터 배열 등을 하나로 묶은 자료형을 말한다.
- 구조체는 연관된 멤버로 구성되는 통합 자료형으로 기존 자료형으로 새로 만들어진 자료형인 유도 자료형이다.
- 구조체를 정의 하기 위해서는 구조체의 틀부터 정의해야 한다.

구조체 선언

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

struct account
{
    char name[12];
    int actnum;
    double balance;
};

int main(void)
{
    struct account mine = { "홍길동", 1001, 300000 };
    struct account yours;

    strcpy(yours.name, "이동원");
    yours.actnum = 1002;
    yours.balance = 500000;

    printf("구조체크기 : %d₩n", sizeof(mine));
    printf("%s %d %.2f₩n", mine.name, mine.actnum, mine.balance);
    printf("%s %d %.2f₩n", yours.name, yours.actnum, yours.balance);

    system("pause");
    return 0;
}
```

구조체 정의

구조체를 선언할 때는 struct를 이용하여 선언한다.
초기화를 할 때는 순서대로 , 로 구분하며 기술하고
아무값도 적지 않는다면 0이 저장된다.

```
구조체크기 : 24
홍길동 1001 300000.00
이동원 1002 500000.00
계속하려면 아무 키나 누르십시오 . . .
```

구조체 변수의 크기는 멤버의 크기의 합보다 크거나 같다

구조체 비교

```
struct student one;
one = bae;
if (one.snum == bae.snum)
    printf("학번이 %d(으)로 동일합니다.₩n", one.snum);
```

- 구조체는 구조체 변수끼리의 비교는 할 수 없지만 구조체 멤버들끼리의 비교는 가능하다.
- 둘 다 구조체 형의 변수일 경우 대입이 가능하다.

```
struct account
{
    struct date open;
    char name[12];
    int actnum;
    double balance;
};
```

- 구조체는 다른 구조체의 멤버를 자신의 멤버로 사용할 수 있다.

공용체

- 공용체는 구조체와 비슷하며 struct 대신 union을 사용한다.

```c
#include <stdio.h>

union data
{
    char ch;
    int cnt;
    double real;
}data1;

int main(void)
{
    union data data2 = { 'A' };
    union data data3 = data2;

    printf("%d %d₩n", sizeof(union data), sizeof(data3));

    data1.ch = 'a';
    printf("%c %d %f₩n", data1.ch, data1.cnt, data1.real);
    data1.cnt = 100;
    printf("%c %d %f₩n", data1.ch, data1.cnt, data1.real);
    data1.real = 3.156759;
    printf("%c %d %f₩n", data1.ch, data1.cnt, data1.real);

    system("pause");
    return 0;
}
```

공용체 정의

공용체는 저장 공간을 공유하므로 마지막에 저장된 값만 저장할 수 있다

```
8 8
a 97 0.000000
d 100 0.000000
N -590162866 3.156759
계속하려면 아무 키나 누르십시오 . . .
```

공용체의 크기는 멤버 중 가장 큰 자료형의 크기로 정해진다

자료형 재정의

- typedef는 이미 사용되는 자료 유형을 다른 새로운 자료형 이름으로 재정의할 수 있도록 하는 키워드이다.
- 재정의 하는 이유는 프로그램의 시스템 간 호환성과 편의성을 위해 필요하다.

```c
#include <stdio.h>

typedef unsigned int budget;

int main(void)
{
    budget year = 24500000;
    typedef int profit;
    profit month = 4600000;

    printf("올 예산은 %d, 이달의 이익은 %d 입니다.₩n", year, month);

    system("pause");
    return 0;
}

void test(void)
{
    budget year = 24500000;
}
```

int를 budget으로 재정의함

```
올 예산은 24500000, 이달의 이익은 4600000 입니다.
계속하려면 아무 키나 누르십시오 . . .
```

구조체 자료형 재정의

- 구조체도 typedef를 이용하여 재정의 할 수 있다.

```c
#include <stdio.h>

struct date
{
    int year;
    int month;
    int day;
};

typedef struct date date;

int main(void)
{
    typedef struct
    {
        char title[30];
        char company[30];
        char kinds[30];
        date release;
    }software;

    software vs = { "비주얼 스튜디오 커뮤니티", "MS", "통합개발환경", {2018, 8, 29} };

    printf("제품명 : %s₩n", vs.title);
    printf("회사 : %s₩n", vs.company);
    printf("종류 : %s₩n", vs.kinds);
    printf("출시일 : %d. %d. %d₩n", vs.release.year, vs.release.month, vs.release.day);

    system("pause");
    return 0;
}
```

struct date를 date로 재정의

date를 멤버로 가지며 구조체를 정의하면서 software로 정의

```
제품명 : 비주얼 스튜디오 커뮤니티
회사 : MS
종류 : 통합개발환경
출시일 : 2018. 8. 29
계속하려면 아무 키나 누르십시오 . . .
```

구조체 포인터

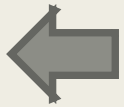
- 구조체 포인터는 구조체의 주소값을 저장하는 변수이다.
- 구조체 포인터도 일반 포인터와 사용법이 비슷하다.
- ->는 멤버 접근 연산자로 구조체 포인터 -> 구조체 멤버로 사용한다.
  (*포인터).멤버와 동일하다.
- *포인터.멤버와는 다르며 이것은 오류가 발생한다.
- *구조체.멤버는 멤버값을 나타내며 한글일 경우 오류가 발생한다.
- *포인터->멤버는 위와 같은 값을 나타낸다.

공용체 포인터

- 공용체 포인터 또한 사용법은 구조체 포인터와 비슷하다.

```c
#include <stdio.h>

int main(void)
{
    union data
    {
        char ch;
        int cnt;
        double real;
    };

    typedef union data udata;

    udata value, *p;

    p = &value;
    p->ch = 'a';
    printf("%c %c₩n", p->ch, (*p).ch);
    p->cnt = 100;
    printf("%d ", p->cnt);
    p->real = 3.14;
    printf("%.2f ₩n", p->real);

    system("pause");
    return 0;
}
```

```
a a
100 3.14
계속하려면 아무 키나 누르십시오 . . .
```



value를 가리키는 포인터 p 선언
p->ch는 value의 ch를 의미
p->ch와 (*p),ch는 동일

구조체 배열

- 구조체도 배열을 선언하여 여러 변수를 저장할 수 있다.

```c
#include <stdio.h>

struct lecture
{
    char name[20];
    int type;
    int credit;
    int hours;
};
typedef struct lecture lecture;

char *lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
char *head[] = { "강좌명", "강좌구분", "학점", "시수" };

int main(void)
{
    lecture course[] = { {"인간과 사회", 0, 2, 2},
    {"경제학개론", 1, 3, 3},
    {"자료구조", 2, 3, 3},
    {"모바일프로그래밍", 2, 3, 4},
    {"고급 C프로그래밍", 3, 3, 4} };

    int arysize = sizeof(course) / sizeof(course[0]);

    printf("배열크기 : %d₩n", arysize);
    printf("₩n%12s  %12s    %6s %6s₩n", head[0], head[1], head[2], head[3]);
    printf("₩n-------------------------------------------------------₩n");

    for (int i = 0; i < arysize; i++)
        printf("₩n%16s %10s %5d %5d₩n", course[i].name,
            lectype[course[i].type], course[i].credit, course[i].hours);

    system("pause");
    return 0;
}
```

```
배열크기 : 5

      강좌명        강좌구분      학점    시수

-------------------------------------------------------

     인간과 사회        교양      2     2

     경제학개론    일반선택      3     3

       자료구조    전공필수      3     3

모바일프로그래밍    전공필수      3     4

고급 C프로그래밍    전공선택      3     4
계속하려면 아무 키나 누르십시오 . . .
```

lecture course[]로 구조체 배열을 선언 및 초기화

배열을 이용하여 멤버로의 접근 또한 가능하다

끝마치며

- 문자와 문자열에서는 c언어의 문자 저장방식에 대해 알 수 있었다. 문자와 문자열에 관한 함수가 많아 처음 강의를 들으며 공부할 때는 많이 헷갈렸지만 이렇게 다시 한 번 정리를 하니 이해하는데 많은 도움이 되었다.
- 변수 유효범위에서는 변수의 종류와 각 특징들을 배울 수 있었고 특히 레지스터는 cpu의 레지스터에 저장된다는 부분에서 컴퓨터 구조시간에 배운 메모리가 도움이 되었다.
- 구조체와 공용체에서는 다양한 값들을 하나의 변수로 묶어 활용하는 것을 배웠지만 포인터와 관련된 부분은 이번에 공부하며 좀 더 이해에 도움이 되었지만 아직 이해는 되지만 헷갈리는 부분이 존재하여 조금 더 공부가 필요함을 느꼈다.

감사합니다