Start

ASCII CODE

- atoi() 구현

```
#include <stdio.h>

int my_toInt(char *str) {
    int num = 0;

    // '\0' 기준으로 문자열의 끝을 확인
    while (*str != '\0') {
        num *= 10;
        num += *(str++) - '0';
    }
    // 12345\0
    return num;
}

void main() {
    // atoi(문자) 숫자로 아스키코드를 이용해서...
    // '0' -> 아스키코드로 48
    // printf("%c, %d\n", '0', '0');
    // printf("%d", '1' - '0');
    char str[10];

    gets(str);
    printf("%s는 변환하면 %d\n", str, my_toInt(str));
}
```

- toupper() 구현

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

char my_toupper(char ch) {
    if (97 <= ch && ch <= 122) {
        return ch - 32;
    }
    return ch;
}

int main(void) {
    char ch[100];
    int i = 0;

    printf("영어 문장을 입력하세요 >> \n");
    gets(ch);
```

```
    for (int i = 0; *ch != '\0'; i++) { // 로직 잘못짬!
        printf("%c", my_toupper(ch[i]));
    }

    return 0;
}
```

명령행 인자

- 프로그램 실행 할 때 넘겨주는 매개변수
- argc : 인자의 개수
- argv : 전달된 인자 (문자열 형태)

구조체

구조체의 이해

- 변수 : 선언된 자료형의 데이터를 담는 상자
- 배열 : 동일한 종류의 데이터 여러개를 담는 여러개의 상자 int [], char [], double [], ...
- 구조체 : 사용자가 자료형들을 모아 하나의 데이터 형식으로 정의해서 담는 상자
    - 내가 원하는 자료형들을 모아서 하나의 자료형으로 정의하는 것. 즉, 사용자 정의 자료형!

구조체 정의

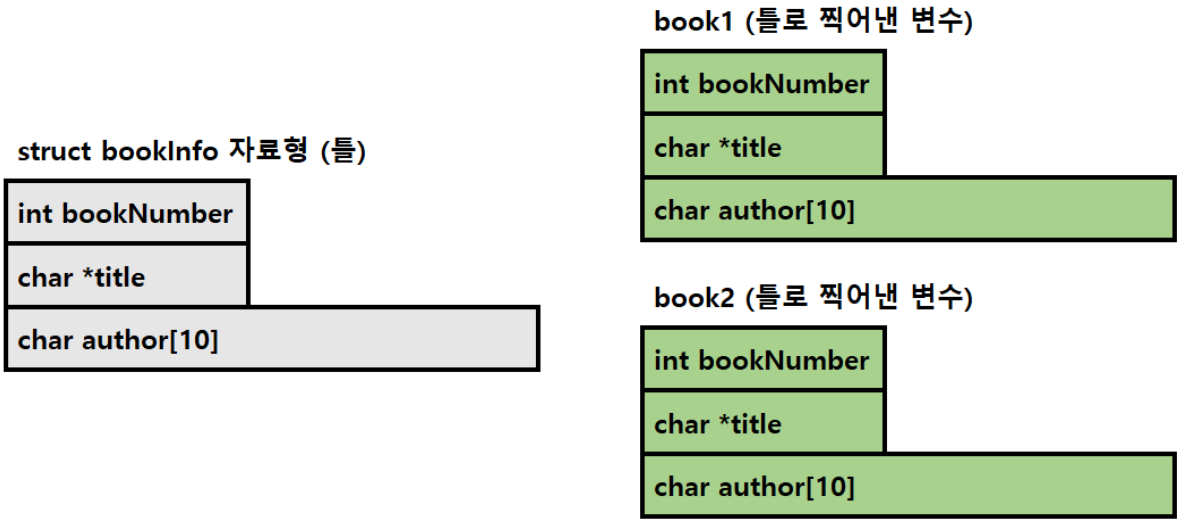
- 사용자 정의 자료형
- typedef (자료형 정의)
- 구조체 멤버

```
struct bookIF { // 기본적으로 자료형은 'struct bookIF' 키워드를 포함한 풀네임으로 써주거나
    int bookNumber;
    char *title;
    char author[10];
}

typedef struct bookInfo {  // typedef를 이용해 자료형을 Book 처럼 축약형으로 만들어줘도 된다.
    int bookNumber;
    char *title;
    char author[10];
} Book;
```

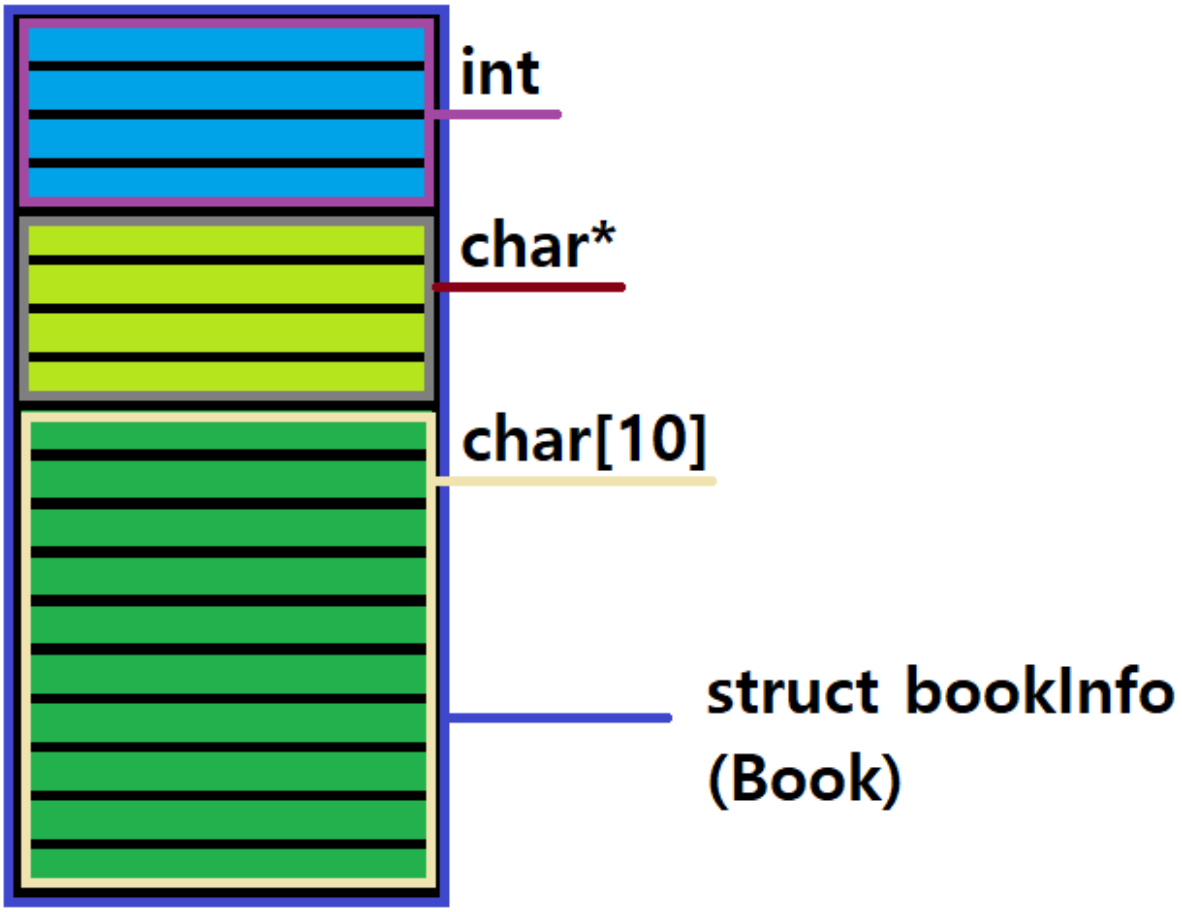
구조체 선언

```c
int main() {
    struct bookIF b1, b2; // 키워드를 포함한 구조체 선언
    Book book1;           // 축약형 이름을 이용한 구조체 선언
    struct bookInfo book2;

    ...
}
```

개념적으로 보면



메모리적으로 보면



구조체 사용

- 멤버변수 접근 : . (도트 연산자)으로 접근
- ~에 속하는. 내장된.. 그런 의미!

```
typedef struct bookInfo {
    int bookNum;
    char *title;
    char author[10];
} Book;

int main() {
    Book book1;

    book1.bookNum = 10; // 내장하고 있는 변수에 접근
    book1.bookTitle  // error! undeclare 정의되지 않은 변수입니다..
    book1.title = "Book name";

    1. scanf("%s", book1.author);       // 둘다
    2. scanf("%s", &book1.author[0]);   // 가능
}
```

간단한 응용

- 이름, 학번, 학점 입력해 저장하기

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

struct position {
    int x;
    int y;
};

void main() {
    struct position p1, p2;
    int xdiffer, ydiffer;
    double distance;

    printf("점의 좌표 입력 (x y): ");
    scanf("%d %d", &p1.x, &p1.y);

    printf("점의 좌표 입력 (x y): ");
    scanf("%d %d", &p2.x, &p2.y);

    xdiffer = p1.x - p2.x;
    ydiffer = p1.y - p2.y;

    distance = sqrt(xdiffer * xdiffer + ydiffer * ydiffer);

    printf("두 점 사이의 거리는 %lf입니다.\n", distance);
}
```

- 두 점 사이의 거리 구하기

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

struct position {
    int x;
    int y;
};

void main() {
    struct position p1, p2;
    int xdiffer, ydiffer;
    double distance;

    printf("점의 좌표 입력 (x y): ");
    scanf("%d %d", &p1.x, &p1.y);

    printf("점의 좌표 입력 (x y): ");
    scanf("%d %d", &p2.x, &p2.y);

    xdiffer = p1.x - p2.x;
```

```
    ydiffer = p1.y - p2.y;

    distance = sqrt(xdiffer * xdiffer + ydiffer * ydiffer);

    printf("두 점 사이의 거리는 %lf입니다.\n", distance);
}
```

- 두 점이 주어질때 면적 둘레 구하기

```
#include<stdio.h>
#include<math.h>

struct point{
    int x;
    int y;
};
struct rect{
    struct point p1;
    struct point p2;
};

void main(){
    struct rect r;
    int w, h, area, round;

    printf("왼쪽 상단의 좌표를 입력하시오: ");
    scanf("%d %d", &r.p1.x, &r.p1.y);

    printf("오른쪽 상단의 좌표를 입력하시오: ");
    scanf("%d %d", &r.p2.x, &r.p2.y);

    w=abs(r.p2.x-r.p1.x);
    h=abs(r.p2.y-r.p1.y);

    area=w*h;
    round=(w+h)*2;
    printf("면적은 %d이고 둘레는 %d입니다.\n", area, round);
}
```

- 구조체 변수의 대입

```
#include<stdio.h>

struct point {
    int x;
    int y;
};

struct point p1 = { 10,20 }; // 전역 변수 선언
struct point p2 = { 30,40 };

void fun();
```

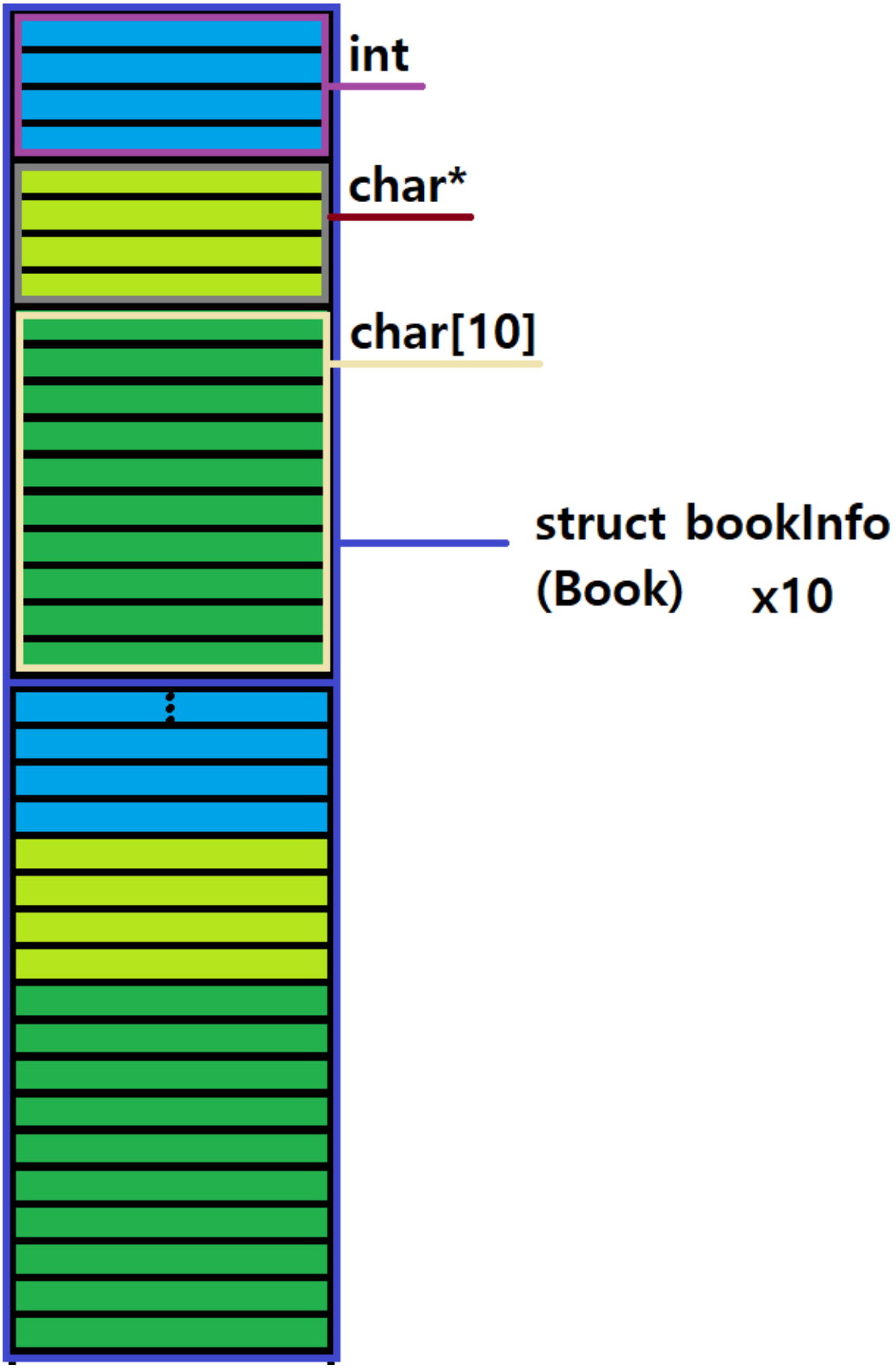
```
void main() {

    p2 = p1;    // 대입 가능. 하나의 자료형이니까!

/*  if(p1==p2)  // But 전체를 비교하는건 불가. 컴파일 오류! */
    if ((p1.x == p2.x) && (p1.y == p2.y)) // 멤버 변수 하나하나 비교해야 함
        puts("p1과 p2가 같습니다.");
}

void fun() {
    p1.x;
}
```

구조체 배열

- 배열 : 동일한 종류의 데이터 여러개를 담는 여러개의 상자
    - int arr[10], char str[10], double avgs[10], ...
- 구조체 배열 : 배열하고 똑같다. 자료형을 사용자 정의 자료형인 구조체로 가질 뿐.
    - struct bookInfo books[10]
    - 또는 Book books[10]

구조체 배열 선언 및 사용

```
typedef struct bookInfo {
    int bookNum;
    char *title;
    char author[10];
} Book;

// 배열하고 다를게 없다. 똑같은 '배열' 이다!!!
BooK b1[10]; // BOOK을 Type으로 가지는 10개짜리 배열

// b1이 의미하는 바는??
b1 == &b1[0];

// 접근 및 사용할 때
b1[0].bookNum = 300;
b1[0].title = "First Book";
strcpy(b1[0].author, "Uncnown");
b1[0].author[2] = 'k';
```


int
char*
char[10]
struct bookInfo
(Book) x10

간단한 응용

- 간이 단어장 만들기

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#define SIZE 3

typedef struct word {
    char eng[20];
    char kor[20];
} WORD;

void main() {
    WORD dict[SIZE];

    for (int i = 0; i < SIZE; i++) {
        printf("영어 단어 입력 >> ");
        scanf("%s", dict[i].eng);
        printf("한글 단어 입력 >> ");
        scanf("%s", dict[i].kor);
    }

    for (int i = 0; i < SIZE; i++) {
        printf("영어 단어 : %s\n한글 단어 : %s\n", dict[i].eng, dict[i].kor);
    }
}
```

구조체 포인터

malloc 할 때 한 번 더 보면 좋을 듯

- 구조체 자료형의 주소를 담는 포인터
    - 포인터 자료형이 각각 달랐듯, '구조체'의 주소를 담을 수 있는 변수이다.
- 마찬가지로 `자료형 * 이름` 형태로 선언한다.
- 멤버변수 접근시 도트연산자 `.` 을 썼었는데, 포인터에서는 `->` 를 사용

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

typedef struct bookInfo {
    int bookNum;
    char* title;
    char author[10];
} Book;

Book* bp; // 선언 포인터 size도 역시 4byte

int main() {
    Book bArr[2] = { {1, "first", "Park"}, {2, "second", "Kim"} };
```

```
    bp = bArr; // == &bArr[0] 구조체 배열의 이름이 주소

    // 구조체 배열 접근
    printf("%d, %s, %s\n", bArr[0].bookNum, bArr[0].title, bArr[0].author);
    printf("%d, %s, %s\n", bArr[1].bookNum, bArr[1].title, bArr[1].author);

    // 포인터 형식 접근
    printf("%d, %s, %s\n", bp->bookNum, bp->title, bp->author); // ->로 접근
    printf("%d, %s, %s\n", (bp + 1)->bookNum, (bp + 1)->title, (bp + 1)-
>author);
    // 인덱스를 더해서 해당 포인터 값에 ->로 접근

}
```

함수와 구조체

함수의 매개변수 인자로 쓰이는 경우

```
#include<stdio.h>

typedef struct bookInfo {
    int bookNum;
    char* title;
    char author[10];
} Book;

// 구조체
void func1(struct bookInfo b1, struct bookInfo b2);
void func2(Book b1, Book b2);

// 구조체 배열
void func3(struct bookInfo books[10]);
void func4(struct bookInfo* books);
void func5(Book books[10]);
void func6(Book* books);
```

- 함수에 구조체 배열, 포인터를 넘겼을 때는 Call by reference 이므로 -> 를 이용해 접근해야 한다!

함수의 반환형으로 쓰이는 경우

```
#include<stdio.h>

typedef struct bookInfo {
    int bookNum;
    char* title;
    char author[10];
```

```
} Book;

// 구조체
struct bookInfo func1();
Book func2();

// 구조체 배열
struct bookInfo* func3();
Book* func4();
```

간단한 응용

- 점수가 높은 학생을 반환하는 프로그램

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


typedef struct student {
    int number;
    char name[20];
    double grade;
} STD;

STD higher(STD* s);

void main() {
    STD s[3] = { {30,"KIM",4.3}, {31,"KIM",4.5}, {32,"MIN",3.7} };
    STD highst;

    for (int i = 0; i < 3; i++) {
        printf("학생의 번호=%d, 이름=%s, 성적=%.1f\n", s[i].number, s[i].name,
s[i].grade);
    }

    puts("");
    highst = higher(s);

    printf("점수가 더 높은 학생은\n학생의 번호=%d, 이름=%s, 성적=%.1f\n",
highst.number, highst.name, highst.grade);
}

STD higher(STD* s) {
    double max = s[0].grade;
    int max_index = 0;

    for (int i = 1; i < 3; i++) {
        if (max < s[i].grade) {
            max = s[i].grade;
            max_index = i;
        }
    }
```

```
    return s[max_index];
}
```

구조체 연습 문제

연습문제 1

- 좌표 평면 두 점의 좌표(x,y)를 입력 받아 저장한 뒤, 두 점을 지나는 선의 기울기를 출력하는 프로그램을 작성하시오.
    - 입력 및 출력 예시

```
0 0
3 3
기울기는 1 입니다.
```

연습문제 2

- 좌표 평면의 두 점 좌표(x,y)를 입력 받아 저장한 뒤, 두 점을 꼭짓점으로 하는 사각형이 정사각형인지 판단하는 프로그램을 작성하시오.
    - 입력 및 출력 예시

```
4 5
1 3
정사각형이 아닙니다.

2 4
4 6
정사각형 입니다.
```

연습문제 3

나만의 단어장 프로그램 구현하기

- 크기 20짜리 영어단어를 담을 변수 eng, 한국어 뜻을 담을 변수 kor를 멤버로 가지는 구조체 dictionary를 정의하라
- typedef를 이용해 축약형 구조체 이름을 DICT로 정의하고, 크기 5의 DICT 배열 words을 전역으로 선언하라
- 현재 단어장에 들어있는 개수를 나타내는 count를 전역변수로 선언하라
- 단어를 입력받아 저장하는 함수 void addWord(); 함수를 구현하라

- 영어 단어를 입력받아 사전에서 찾아서 출력하는 함수 void findWord(); 함수를 구현하라
- 현재 단어장의 전체 내용을 출력하는 void printWords(); 함수를 구현하라
- main 함수에서 반복문을 이용해 적절히 호출하여 프로그램을 완성하라