1) 컴파일링

컴파일링 이란?

개발자가 작성한 코드를 컴퓨터 언어 (0,1)로 바꾸는 과정 C 코드를 실행하기 전, 컴파일링 과정이 필요하다.

컴파일링의 4 단계

전처리 : 컴파일 전 무언가 실행하라고 알려줌 - 라이브러리에 파일 추가(#include <stdio.h>)

컴파일링 : 소스코드를 어셈블리 코드(저수준의 프로그래밍 언어)로 변환

어셈블링 : 어셈블리 코드를 오브젝트 코드(0,1)로 변환 링킹 : 여러개의 파일을 하나의 오브젝트 파일로 합침

컴파일링 하는 법

C 언어 : make 파일이름

2) 디버깅

디버깅 이란?

코드 속 버그를 식별하고 고치는 과정

문법적 오류

컴파일링이 안됨

ex) #include <stdio.h>를 빼먹음

컴파일링 할 때 help50 make 파일이름 을 입력하면 도움을 받을 수 있음

논리적 오류

컴파일링은 되는데 원하는 답이 안나옴

브레이크 포인트 지정 -> 컴파일링 -> debug50 ./파일이름 을 하면 브레이크 포인트 부터 한 줄씩 코드가 실행됨

디버깅 종료: ctrl+c

3) 코드의 디자인

코드의 디자인이 중요한 이유?

여러사람이 함께 작업할 때, 서로의 코드를 쉽게 이해하려면 디자인의 통일이 필요하다.

디자인 검사 프로그램

style50 hello.c 를 넣으면, 코드 디자인의 어느 부분이 수정이 필요한지 표시된다.
ex) { printf ("hello"); } -> { printf ("hello"); }

4) 배열 (1)

배열 이란?

연관된 데이터를 모아서 관리하는 데이터 타입 연관된 데이터를 메모리(RAM)에 연이어 저장하기 위해 사용된다.

```
ex)   int scores[3];
   scores[0] = 72;
   scores[1] = 73;
   scores[2] = 33;
```

메모리(램)에 저장되는 방법

H (1bite)	E	L	L
O			

bool: 불리언, 1 바이트
char: 문자, 1 바이트
int: 정수, 4 바이트
float: 실수, 4 바이트
long: (더 큰) 정수, 8 바이트
double: (더 큰) 실수, 8 바이트
string: 문자열, ?바이트

5) 배열 (2)

전역변수

전역변수는 전역에서 사용할 수 있는 변수로 바깥에 적어준다.
반면, 지역변수는 {} 안에서만 사용 가능하다.

`const int N = 3;` // 전역변수, 대문자로 써주는 게 관례

```
const int N = 3; //전역변수, 대문자로 써주는 게 관례

int main(void)

{

   int scores[N];

    scores[0] = 72;

    scores[1] = 73;

    scores[2] = 33;


    // 평균 점수 출력

    printf("Average: %i\n", (scores[0] + scores[1] + scores[2]) / N);

}
```

루프함수를 이용한 배열의 동적 활용

```
#include <cs50.h>

#include <stdio.h>


float average(int length, int array[]);
```

```c
int main(void)
{
    // 사용자로부터 점수의 갯수 입력
    int n = get_int("Scores:  ");

    // 점수 배열 선언 및 사용자로부터 값 입력
    int scores[n];
    for (int i = 0; i < n; i++)
    {
        scores[i] = get_int("Score %i: ", i + 1);
    }

    // 평균 출력
    printf("Average: %.1f\n", average(n, scores));
}
//평균을 계산하는 함수
float average(int length, int array[])
{
    int sum = 0;
    for (int i = 0; i < length; i++)
    {
        sum += array[i]; //이해가 안되는 부분***
    }
    return (float) sum / (float) length;
}
```

6) 문자열과 배열

널 종단 문자

문자열의 끝을 나타내는 문자 (\0) ex) string a = "HI!";

H (a[0])	**I** (a[1])	**!** (a[2])	**\0** (a[3])

문자열의 배열

```
string names[4];



names[0] = "EMMA";

names[1] = "RODRIGO";

names[2] = "BRIAN";

names[3] = "DAVID";



printf("%s\n", names[0]);

printf("%c%c%c%c\n", names[0][0], names[0][1], names[0][2], names[0][3]);
```

names[0][0] = "E" ('E'MMA)
names[1][1] = "O" (R'O'DRIGO)
names[2][2] = "I" (BR'I'AN)
names[3][3] = "I" (DAV'I'D)

E	M	M	A	\0

R	O	D	R	I
G	O	\0	B	R
I	A	N	\0	D
A	V	I	D	\0

7) 문자열의 활용

문자열의 길이

문자열의 길이를 알려주는 함수: strlen (#include string.h)

```
#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get string("Input: ");
    printf("Output:\n");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        printf("%c\n", s[i]);
    }
}
```

대문자로 바꾸기

대문자로 바꿔주는 함수: toupper (#include ctype.h)

```
#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get string("Before: ");
    printf("After:  ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        printf("%c", toupper(s[i]));
    }
    printf("\n");
}
```

8) 명령행 인자

명령행 인자 란?

컴파일 후 저장하고자 하는 파일명과 같이 추가적인 정보를 함께 줄 수 있는데, 이런 정보를 명령행 인자라 한다.

```
#include <cs50.h>
#include <stdio.h>


int main(int argc, string argv[])
{
```

```
    if (argc == 2)
    {
        printf("hello, %s\n", argv[1]);
    }
    else
    {
        printf("hello, world\n");
    }
}
```

위 파일을 arg.c 로 저장, 컴파일링 한 후
./arg 를 하면 hello, world 가 나온다.
./arg david 를 하면 hello, david 가 나온다.

argc 는 받게 될 입력의 갯수로 [0] 파일이름 [1] david => 총 2 개
argv 는 argv[0]: 파일이름 argv[1]: david 가 된다.

만약, argc == 3 printf(~~, argv[2])
./arg david anna 를 하면 hello, anna 가 나온다.