CSE2010
자료구조론 실습
Week 1: C Review “From Basic To Pointer”

한양대학교 ERICA
소프트웨어융합대학
ICT 융합학부

공지

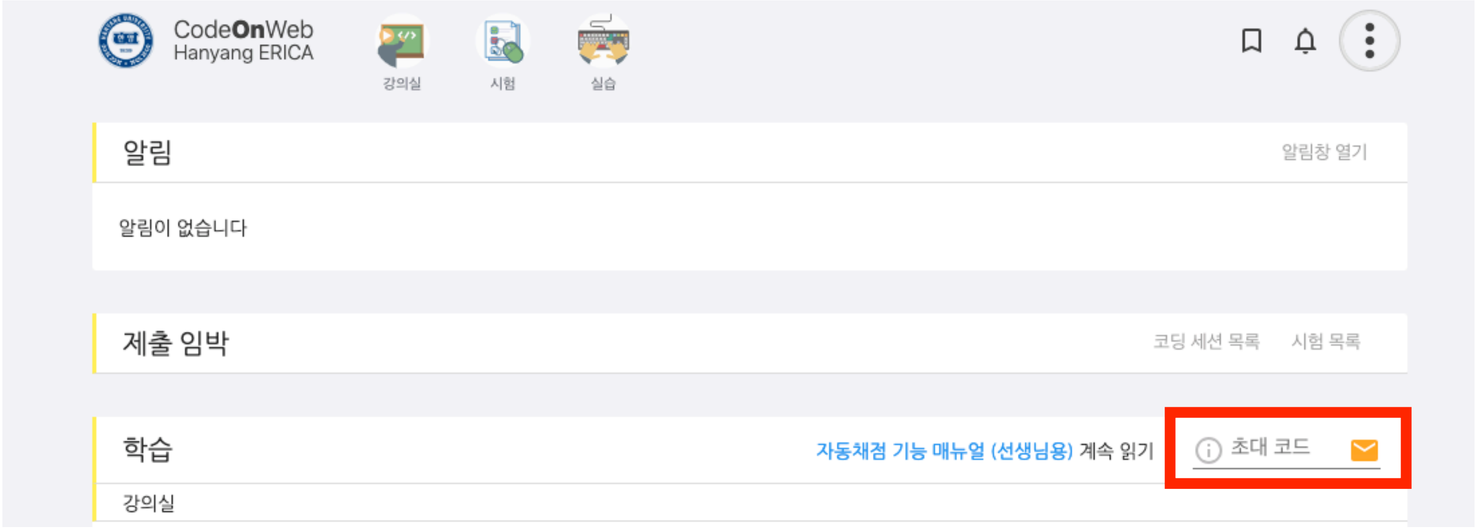
- 실습조교
  - **윤지우** 석사과정(yoonjeewoo@gmail.com)
  - 양미경 ICT 융합학부 오전반 오전 11시 ~
  - 최준두 ICT 융합학부 오후 A반 오후 1시 ~
  - 정승훈 ICT 융합학부 오후 B반 오후 1시 30분 ~
- 실습문제
  - 첫째 (C 언어 기초) 주를 제외한 나머지 수업들은 모두 1~2 문제 입니다.
  - 난이도는 피드백을 통해 조절 => **윤지우** 조교에게 메일을 보내주세요!
  - 배점
    - 미제출: 0점
    - 완성제출: 1점 (테스트 케이스를 모두 통과하면!)
  - 제출기한: 실습 당일 23:59분 ! 단 실습실은 문제를 모두 풀어야 퇴실 하실 수 있습니다.
  - **에러를 두려워하지 말기!** => 시간이 있으니 뭐라고 써 있는지 천천히 읽어보세요 :)
  - **코드는 자신이 직접 짜기!**

오늘의 실습

- **CodeOnWeb** 사용법
- C 언어 기초 훑어보기(가장 많이 쓰일 것들 위주로…)
  - **변수, 입출력**
  - **헤더파일**
  - **조건문**
  - **반복문**
  - **배열**
  - **함수**
  - **포인터**
- **배열, 구조체, 포인터 …** 등은 2주차에 한번 더 배웁니다!
- 실습문제는 총 **3문제**

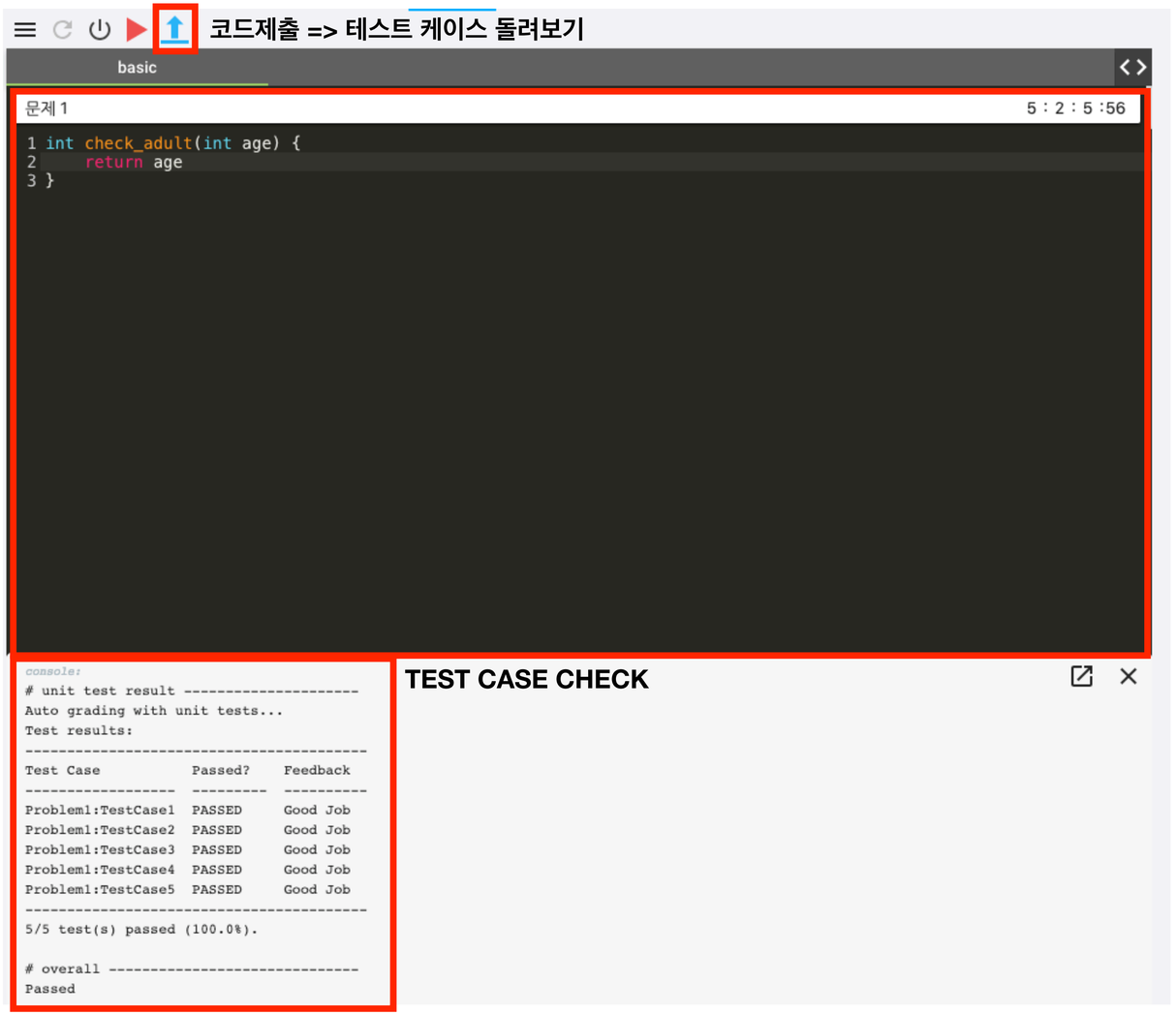
CodeOnWeb

- 온라인 코드 채점 플랫폼
- 주어진 형식에 맞춰서 코드를 작성하면 미리 만들어놓은 테스트 코드가 여러분의 제출코드를 자동채점하게 됩니다!
- https://erica.codeonweb.com 이곳에 들어가서 우선 회원가입을 해주세요!
- 주어진 초대코드를 입력하시면 해당 실습반으로 들어가게 됩니다.



실습 코드 제출방법

TEXT EDITOR




```
# unit test result ----------
Auto grading with unit tests...
Test results:
----------
Test Case            Passed?    Feedback
-----------------    ---------  ----------
Problem1:TestCase1   PASSED     Good Job
Problem1:TestCase2   PASSED     Good Job
Problem1:TestCase3   PASSED     Good Job
Problem1:TestCase4   PASSED     Good Job
Problem1:TestCase5   PASSED     Good Job
----------
5/5 test(s) passed (100.0%).

# overall ----------
Passed
```

변수와 입출력

- **변수**
  - 데이터(숫자, 문자, 구조체 등)를 담을 공간.
  - 변수를 선언한다 => 데이터를 저장할 상자를 만든다!
- **입력**과 **출력**
  - **입력**은 내가 키보드에 입력하는 것을 컴퓨터에 보내는 것.
  - **출력**은 컴퓨터에 저장되어 있는 값을 출력하는 것.
  - **입력**은 `scanf("[%변환문자]",&[변수])`
  - **출력**은 `printf("[출력하고 싶은 문자열 + %변환문자]", [변수])`
  - **변환문자**의 개수와 **변수**의 개수는 반드시 일치해야함.
  - 여러개의 **입력**을 받고싶은 경우, 띄어쓰기를 통해서 구분가능.
    - `scanf("%d %d %d", &a, &b, &c);`
  - **출력**에서 사용하는 변환문자는 수업자료 참조.

```c
#include <stdio.h> // 준비하기

int main(void) { // 메인 함수 선언하기

    char name[10], dept[10]; // %s 문자열
    int age; // %d 정수
    float height; // %f 실수

    scanf("%s %s %d %f", name, dept, &age, &height);

    printf("name: %s\n", name);
    printf("age: %d\n", age);
    printf("department: %s\n", dept);
    printf("height: %.1f\n", height); // %.1f 는 소수점 아래 첫째 자리까지 출력

    return 0; // 끝내기
}
```

헤더파일과 #define

- **헤더파일**
  - “main.c” 가 실행되기 전에 필요한 요소들을 따로 저장 해놓고 불러오고 싶을 경우
  - “header.h” 파일을 생성하여 미리 코드를 작성한 후, “main.c” 의 상단부분에서 `#include “header.h”` 를 통해서 불러온다.
  - `#include <stdio.h>` 도 헤더파일을 미리 부르는 것!
- **#define**
  - `#define` 을 사용하게 되면, 데이터를 일반 글자로 대체하는 것이 가능하다.
  - 코드의 가독성을 높이고, 유지/보수가 쉬워진다.

```c
#include <stdio.h> // "stdio.h" 파일 호출은 여기에!

/*
    #define 이라는 코드를 사용하여 미리 변수에 값을 정해놓을 수 있음!
    Q. 장점?
    A. 코드의 수정/보완에 좋다. 가독성이 좋아진다.
*/

#define AGE 25
#define DEPT "HCI"
#define HEIGHT 183.5
#define NAME "Jeewoo"
```

define.h

```c
#include "define.h"

int main(void) {

    /*
       NAME, AGE, DEPT, HEIGHT 는 모두 "define.h" 라는 헤더파일에서 온 변수!
       Q. "stdio.h" 는 어디로 갔을까?
    */
    printf("name: %s\n", NAME);
    printf("age: %d\n", AGE);
    printf("department: %s\n", DEPT);
    printf("height: %.1f\n", HEIGHT);

    return 0;
}
```

define.c

조건문

- 말그대로 논리의 흐름에 **조건**을 걸어서 **분기**를 나눌 때 사용함.
  - `if (condition 1) { // something happens here! }`
  - `else if (condition 2) { // if not? here! }`
  - `else { // and here! }`
- 기본적으로 위의 형태를 따르며, else if 구간은 여러번 사용할 수 있음!
- **하지만, if**와 **else**는 각 섹션에 하나씩만 사용! (병렬로 사용하는 것은 상관없음!)

```c
#include <stdio.h>

int main(void) {

    int age; // 변수 선언

    printf("당신의 나이를 입력하세요!");
    scanf("%d", &age); // 나이 입력
    printf("당신의 나이는 %d 세 입니다.", age);

    if (age < 20 ) { // 만약 나이가 20세 미만이라면?
        printf("당신은 미성년자 입니다.\n");
    } else { // 그렇지 않다면?
        printf("당신은 성인 입니다.\n");
    }

    return 0;
}
```

문제 1

- 함수명: `int check_adult(int age);`
- 나이를 함수인자로 받아서
  - 나이가 20살 미만이면 0
  - 나이가 20살 이상이면 1
  - 나이가 음수면 2를 리턴하는 함수를 만드시오.

```c
int check_adult(int age) {
  //insert code here!!
}
```

```c
#include <stdio.h>

int check_adult(int age);

int main(void) {

  printf("%d\n", check_adult(20));
  printf("%d\n", check_adult(2));
  printf("%d\n", check_adult(-1));

  return 0;

}

```

```
➜ week1 ./a.out
1
0
2
```

반복문

- 특정한 코드를 **반복**해서 실행하기 위한 장치
- 반복의 횟수와 조건은 미리 정해줘야 한다.
- For Loop
  - **초기화**, **진행조건**, **변화**로 나누어 생각
  - 초기화: i = 0 과 같은 값으로 i 의 시작 값을 정해줌.
  - 진행조건: i < 10 과 같은 형태로 i 가 10보다 작을 경우 진행할 수 있도록 제한
  - 변화: i++ 와 같은 형태로 반복문의 가장 마지막에 i 의 값을 업데이트 해줌.
- While Loop
  - **진행조건**만 존재한다.
  - **진행조건**이 **True**일 경우 계속 반복 (무한루프도 가능)

```
for (start; condition; update) {
  // something happens here!
}

while (condition) {
  // something happens here!
}
```

```c
#include <stdio.h>

int main() {

    int num; // 입력을 받기 위한 변수
    int sum = 0; // 총 합을 누적하기 위한 변수

    for (int i=0;i<10;i++) {
        scanf("%d", &num);
        sum += num; // 누적
    }

    printf("총 합은 %d 입니다.\n", sum);
}
```

배열

- **배열**
  - **변수**가 하나의 데이터를 담는 **상자**라면, **배열**은 그 상자들의 **모음집**.
  - 선언을 할 때, **배열의 크기**를 정해준다! (초기화를 해준다면 생략해도 상관없음!)
  - 문자열은 캐릭터 변수들의 **배열**!
  - `int a[10]; // a라는 이름의 정수형 원소 10개를 저장할 수 있는 배열 선언하기`
  - `int a[4] = {1, 3, 4, 5} // 선언과 동시에 값 지정도 가능`
  - 배열의 원소에 접근하는 방법: `a[원소의 인덱스]`
  - 예시)

```
int a[4] = {1, 2, 3, 4};
printf("%d", a[2]); // 3 출력
printf("%d", a[4]); // error! why?
// 배열의 인덱스는 0부터 시작! 따라서 a[4] 는 존재하지 않음!
```

  - C언어에 문자열 자료형은 없음! 파이썬, 자바와 다름!
  - char형 배열을 만들어 문자열로 대체! 첫번째 예제의 이름과 학부 입력 참조!

```c
#include <stdio.h>

int main() {

  int a[5] = {1, 2, 3, 4, 5};

  for (int i=0;i<5;i++) {
    printf("%d\n", a[i]);
  }

}
```

문제 2

- 함수명: `int get_max(int arr[]);`
- 정수형 배열을 받아서 그 중 최대값을 리턴하는 함수를 구현하시오.
- 배열의 크기는 5로 한정한다.
- 숫자는 양의 정수라고 가정한다.

함수

- 하나 혹은 여러 개의 값을 받아서 하나 혹은 여러 개의 작업을 수행하는 하나의 장치라고 생각.
- 함수의 형태
  - 자료형_of_something 함수이름(함수인자) {
    ```
    // something happens here
    return something;
    ```
    }
- 예를 들어, 주어진 두 수를 더하는 함수를 만든다고 하면
  - ```
    int add(a, b) {
        return a + b;
    }
    ```
  - `add(1, 2); // 3 반환`
  - `printf("%d", add(1, 2)); // 3 출력`
  - 이 두 가지의 차이를 명확하게 구분 !!

```c
#include <stdio.h>

int add(int a, int b) {
  return a+b;
}
int sub(int a, int b) {
  return a-b;
}
int mul(int a, int b) {
  return a*b;
}
int div(int a, int b) {
  return a/b;
}

int main() {

  int n1, n2;
  scanf("%d %d", &n1, &n2);

  printf("%d\n", add(n1, n2));
  printf("%d\n", sub(n1, n2));
  printf("%d\n", mul(n1, n2));
  printf("%d\n", div(n1, n2));

  return 0;
}
```
```

포인터

- **포인터**는 메모리의 **특정위치**를 가리킬 때 사용한다.
- 기본적인 형태는 다음과 같다.
  - 자료형 *포인터이름 => `int *pointer`
  - 포인터이름 = &변수(변수의 주소)
- 포인터에는 변수의 주소가 저장이 된다.
- *포인터를 하게 되면 포인터가 가리키고 있는 주소에 할당된 값을 얻을 수 있다.

```c
#include <stdio.h>

int main() {

  int a = 20;
  int *pointer_of_a;
  int **pointer_of_pointer_of_a;

  pointer_of_a = &a;
  pointer_of_pointer_of_a = &pointer_of_a;

  printf("포인터가 가리키고 있는 값: %d\n", *pointer_of_a);     // 20
  printf("변수 a의 주소: %p\n", &a);

  a = 30;

  printf("포인터가 가리키고 있는 값: %d\n", *pointer_of_a);
  printf("포인터가 가리키는 주소: %p\n", pointer_of_a);

  printf("포인터의 포인터가 가르키고 있는 값: %d\n", **pointer_of_pointer_of_a);
  printf("포인터의 포인터가 가르키고 있는 주소: %p\n", *pointer_of_pointer_of_a);
}
```

문제 3

- 함수명: `void swap(int *, int *);`
- 함수의 인자로 두 숫자 i와 j의 주소를 입력받아 두 값을 서로 교환한다.

```c
#include <stdio.h>

void swap(????);

int main(void){

  int i = 8, j = 5;
  swap(????);
  printf("%d %d\n", i, j);

  return 0;
}

void swap(????) {

}
```

수고하셨습니다.

- 실습 수업에 대한 문의사항은 yoonjeewoo@gmail.com 으로 메일을 보내주세요!