Review

- Buffered Character I/O
  - putchar
  - getchar
- Formatted I/O
  - printf
  - scanf
- I/O from Files
  - fputc
  - fgetc
  - fprintf
  - fscanf

Structures

Lecture 33

Hyung-Sin Kim

SNU Graduate School of Data Science

Structures

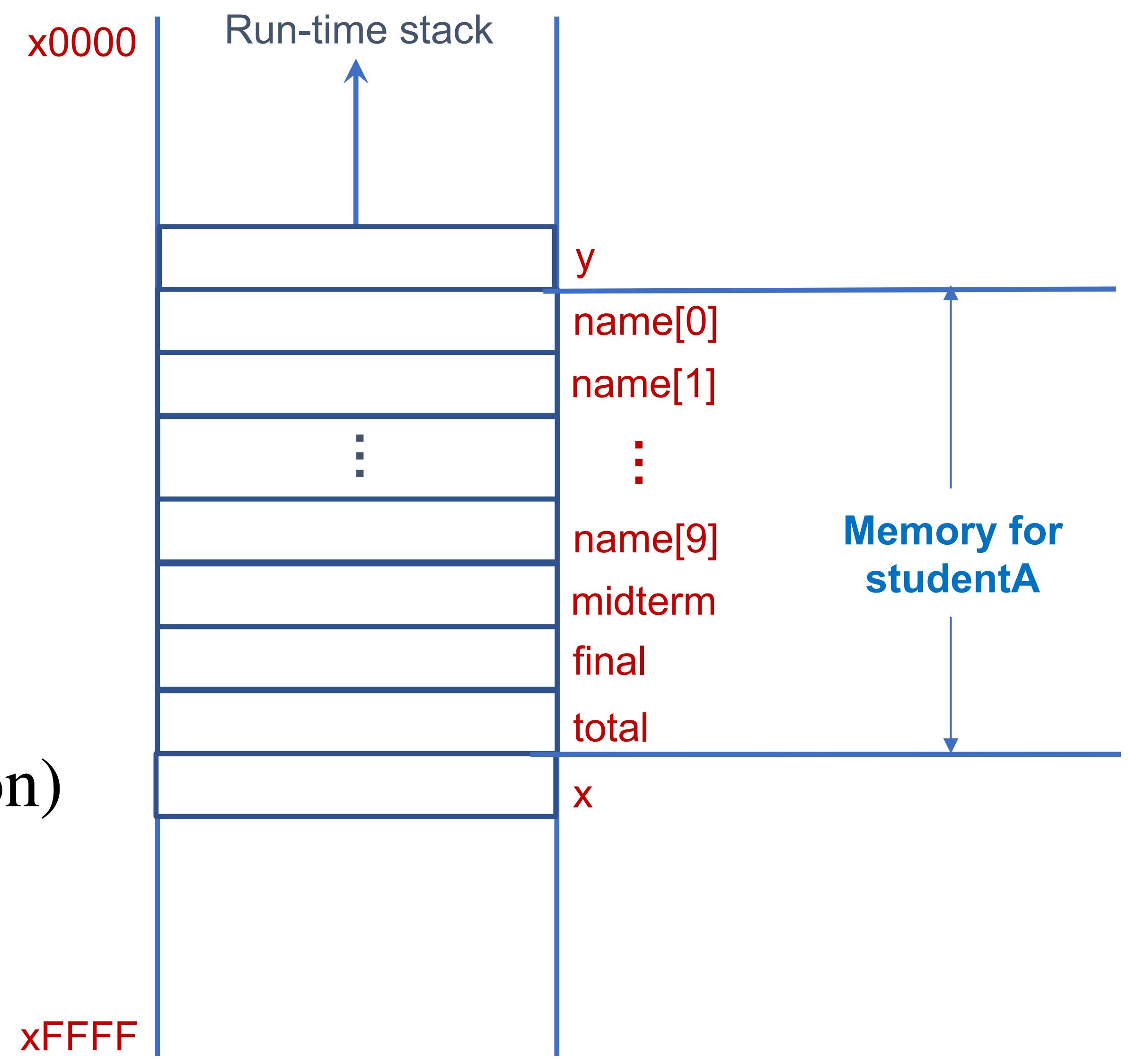
- A convenient way of representing objects that are best represented by combinations of the basic data types
  - For example, if there are many characteristics of a student, such as name, midterm, final, and total, we can declare a **single** memory object (i.e., **structure**) that represents a **student**

- Definition – a studentType structure comprising 4 **members**
  - struct studentType {
  -     char name[10];
  -     int midterm;
  -     int final;
  -     int total;
  - };

Similar to but different from **class** in that it does not have **methods** (class is in C++ ☺)

Structures

- Declaration
  - struct studentType studentA;
- Accessing members (dot operator)
  - studentA.name = “inhoe”;
  - studentA.midterm = 100;
  - studentA.final = 100;
- Memory allocation (contiguous region)
  - int x;
  - struct studentType studentA;
  - int y;



Structures – typedef

- C structures enable programmers to define their own aggregate types
  - typedef <type> <name>;
  - You can use “name” instead of “type” later
  - A convenient way of programming

- Examples
  - typedef int intNum;
    - Now there is a data type “intNum,” which is synonymous with integer
    - intNum valA; declares variable valA whose type is intNum
  - Typedef struct studentType Student;
    - Now there is a data type “Student,” which is synonymous with struct studentType

Structures – Arrays and Pointers

- C provides arrays of structures
  - Student s[5];   // s[0], s[1], s[2], s[3], and s[4] are all structures

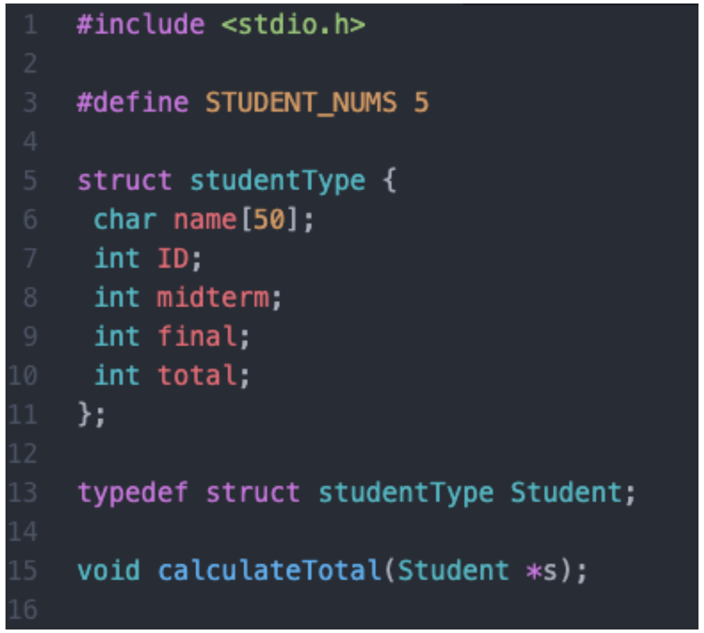
- C provides pointers for structures
  - Student s;
  - Student *sPtr = &s;
  - Member access
    - (*sPtr).midterm   or   sPtr->midterm
    - (*sPtr).final   or   sPtr->final

Practice – Grading System Again (Array version)

```
#include <stdio.h>
#define STUDENT_NUMS 5

int main(void) {
   int midterm[STUDENT_NUMS];
   int final[STUDENT_NUMS];
   int total[STUDENT_NUMS];

  // Input exam scores
  for (int i=0; i < STUDENT_NUMS; i++) {
      printf("Input midterm score for student %d: ", i);
      scanf("%d", &midterm[i]);
      printf("Input final score for student %d: ", i);
      scanf("%d", &final[i]);
   }

  // Calculate total scores
  for (int i=0; i < STUDENT_NUMS; i++) {
      total[i] = midterm[i] + final[i];
   }

   // Output the total scores
   for (int i=0; i < STUDENT_NUMS; i++) {
      printf("Total score for Student %d is %d\n", i, total[i]);
   }

   return 0;
}
```

Practice – Grading System Again (Structure version)

- Assume that name is a single string (no empty space)

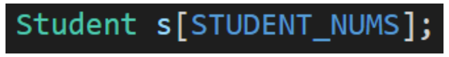


```c
#include <stdio.h>

#define STUDENT_NUMS 5

struct studentType {
 char name[50];
 int ID;
 int midterm;
 int final;
 int total;
};

typedef struct studentType Student;

void calculateTotal(Student *s);

```

```
int main(void) {
    // Declare an array of structures
    /* Your code */

    // Receive input from the keyboard for each student
    /* Your code */

    // Calculate total score (sum) of each student
    /* Your code */

    // Print each student's total score
    /* Your code */
    return 0;
}

// Define calculateTotal
void calculateTotal(Student *s) {
    /* Your code */
}
```

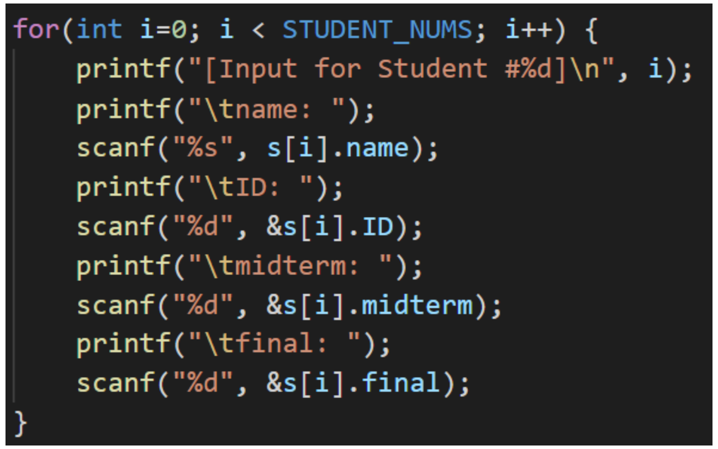
Practice – Grading System Again (Structure version)

- int main(void) {
- // Declare an array of structures



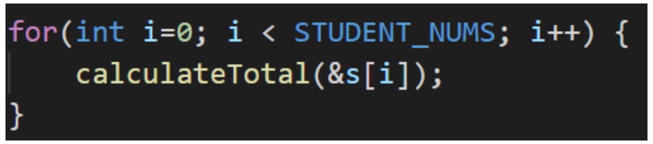
```
Student s[STUDENT_NUMS];
```

- // Receive input from the keyboard for each student



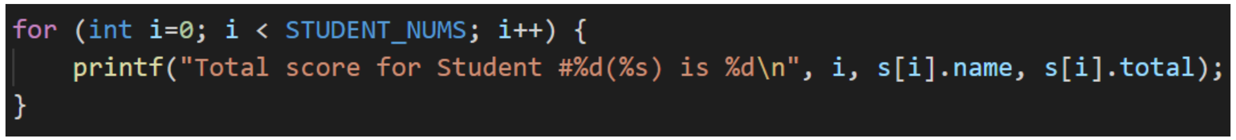
```
for(int i=0; i < STUDENT_NUMS; i++) {
    printf("[Input for Student #%d]\n", i);
    printf("\tname: ");
    scanf("%s", s[i].name);
    printf("\tID: ");
    scanf("%d", &s[i].ID);
    printf("\tmidterm: ");
    scanf("%d", &s[i].midterm);
    printf("\tfinal: ");
    scanf("%d", &s[i].final);
}
```

- // Calculate total score (sum) of each student



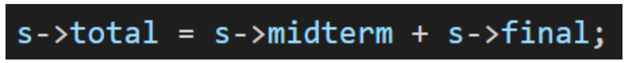
```
for(int i=0; i < STUDENT_NUMS; i++) {
    calculateTotal(&s[i]);
}
```

- // Print each student's total score



```
for (int i=0; i < STUDENT_NUMS; i++) {
    printf("Total score for Student #%d(%s) is %d\n", i, s[i].name, s[i].total);
}
```

- return 0;
- }
- // Define calculateTotal
- void calculateTotal(Student *s) {



```
s->total = s->midterm + s->final;
```

- }

Summary

- Structures
  - Declaration
  - typedef
  - Arrays and Pointers