Structures

Hyung-Sin Kim

Computing Foundations for Data Science

Graduate School of Data Science, Seoul National University

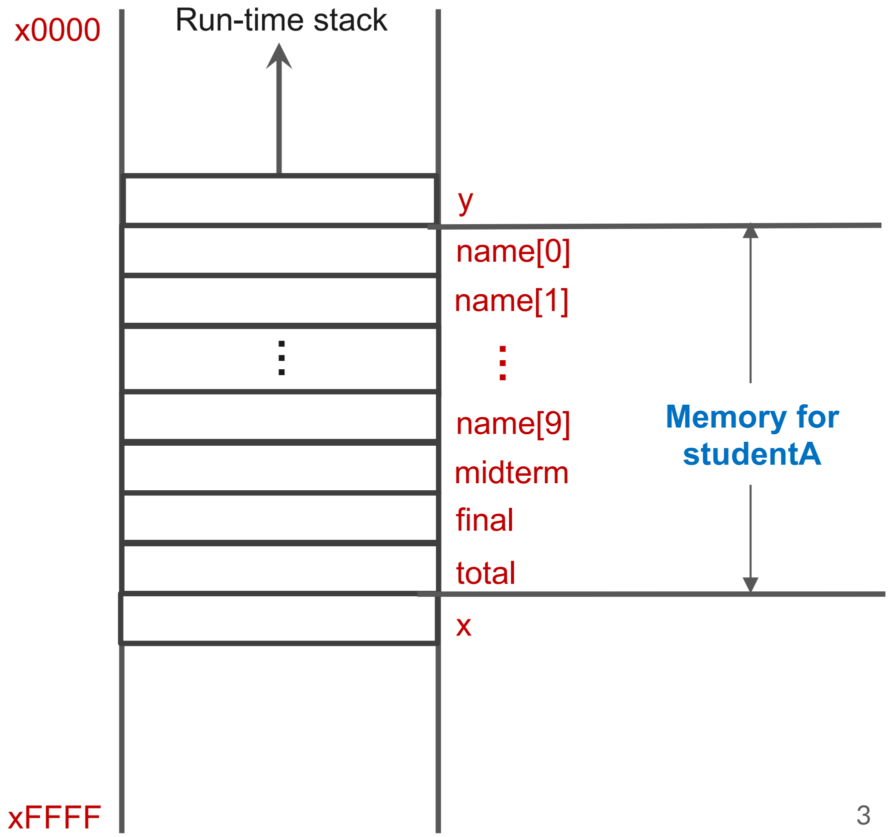
Structures

- 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;
    - };

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>;
    - A convenient way of programming

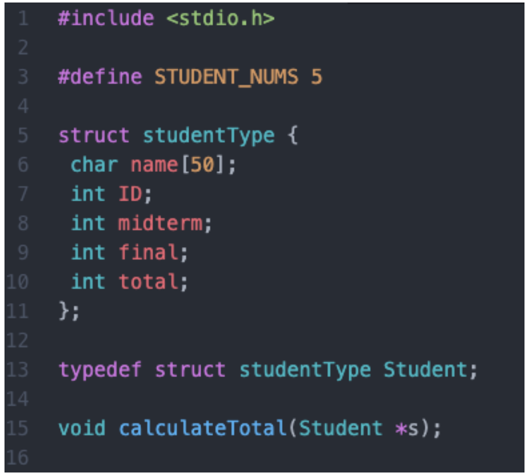
- Examples
    - typedef int intNum;
        - Now there is a data type “intNum,” which is synonymous with integer
        - intNum valA; declares variable valA whose type is intNum
    - Typedef struct studentType Student;
        - Now there is a data type “Student,” which is synonymous with struct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

Structures – Arrays and Pointers

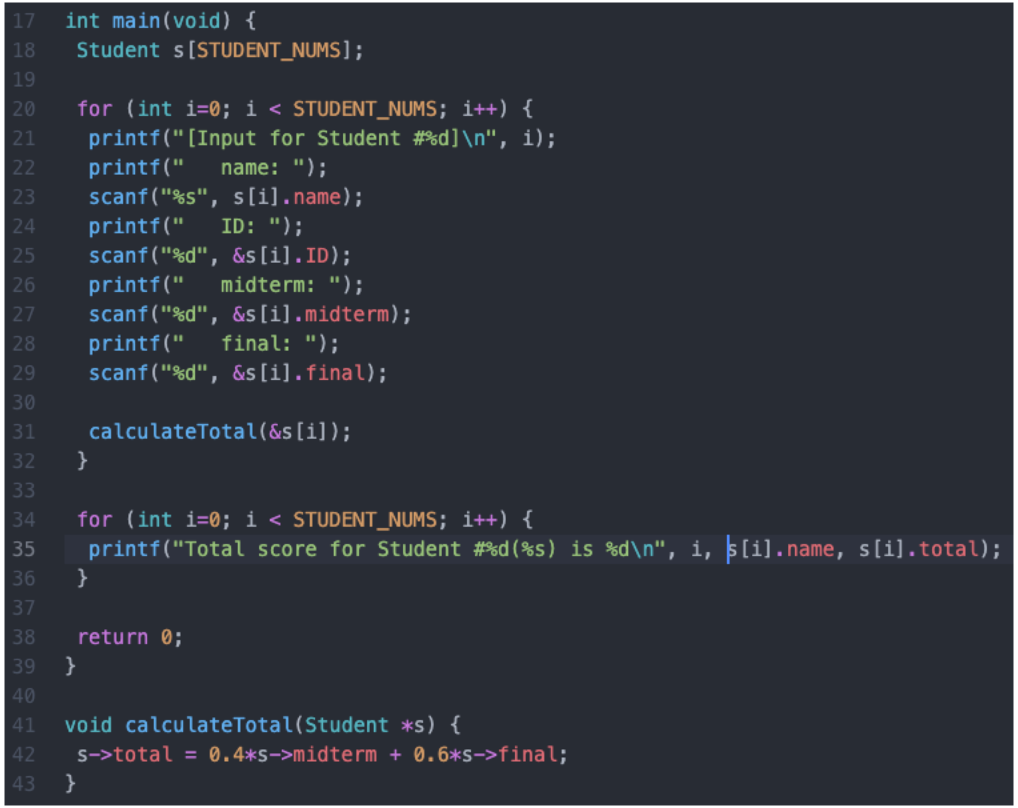
- C provides arrays of structures
  - Student s[5];   // s[0], s[1], s[2], s[3], and s[4] are all structures

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

Example – Grading System (Let’s do it together!)



```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);

```



```c
int main(void) {
 Student s[STUDENT_NUMS];

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

  calculateTotal(&s[i]);
 }

 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;
}

void calculateTotal(Student *s) {
 s->total = 0.4*s->midterm + 0.6*s->final;
}
```

Questions?

Thanks!