CS2100 COMPUTER ORGANISATION

http://www.comp.nus.edu.sg/~cs2100/

Lecture #5c

Arrays, Strings and Structures

NUS National University of Singapore | School of Computing

Questions?

IMPORTANT: DO NOT SCAN THE QR CODE IN THE VIDEO RECORDINGS. THEY NO LONGER WORK

Ask at
https://sets.netlify.app/module/676ca3a07d7f5ffc1741dc65

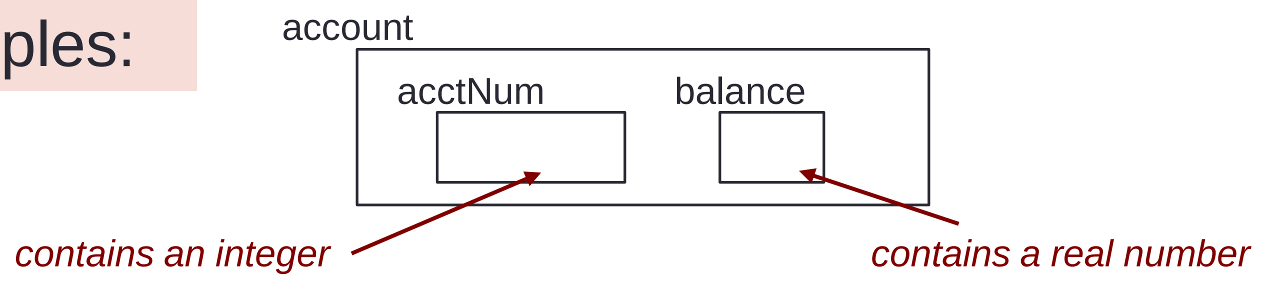
OR

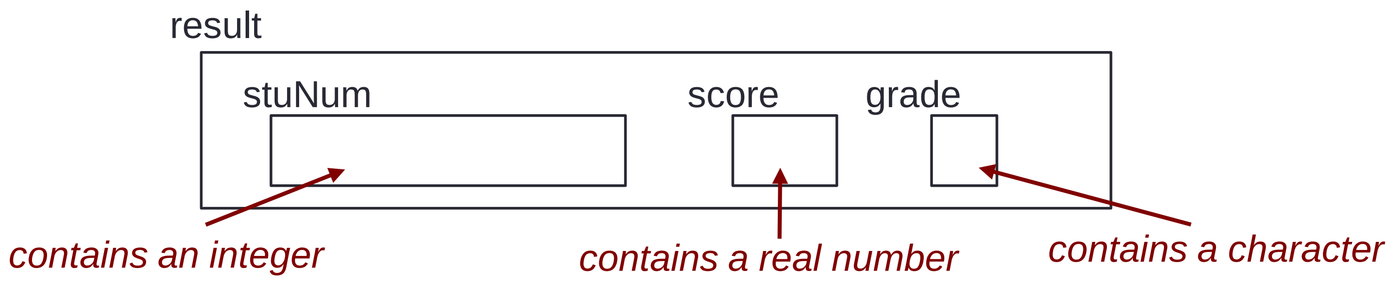
Scan and ask your questions here!
(May be obscured in some slides)

4. Structures (1/2)

- Arrays contain homogeneous data (i.e. data of the same type)
- Structures allow grouping of heterogeneous members (of different types)

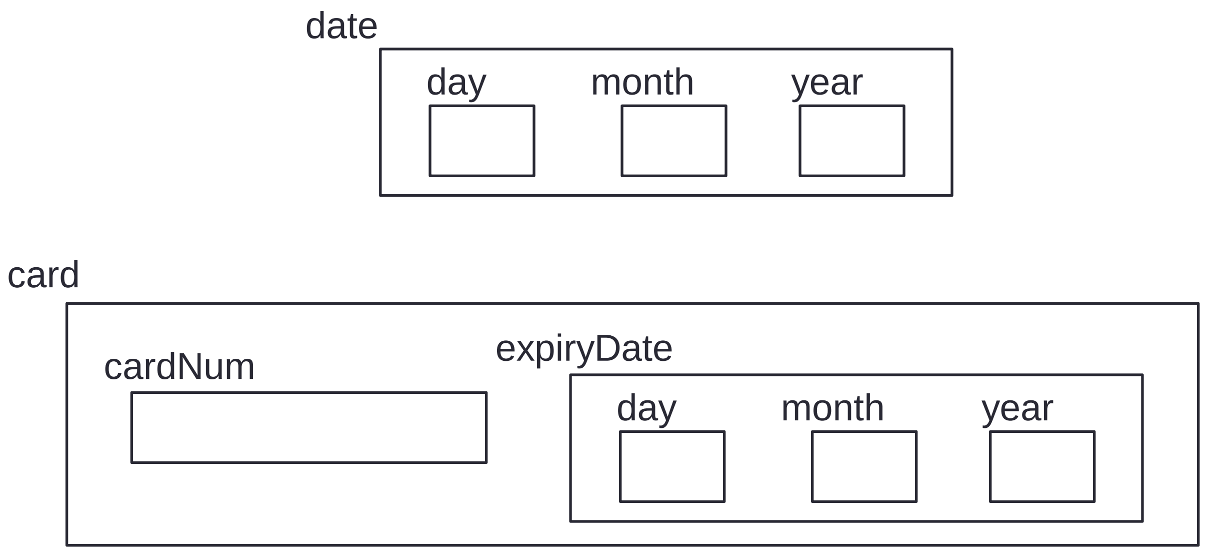
Examples:





4. Structures (2/2)

- A *group* can be a member of another *group*.
- Example: the expiry date of a membership card is of "date" group



4.1 Structure Types (1/2)

- Such a group is called structure type
- Examples of structure types:

```
typedef struct {
   int length, width, height;
} box_t;
```

This semi-colon ; is very important and is often forgotten!

Create a new type called box_t

```
typedef struct {
   int acctNum;
   float balance;
} account_t;
```

Create a new type called account_t

```
typedef struct {
   int stuNum;
   float score;
   char grade;
} result_t;
```

Create a new type called result_t

4.1 Structure Types (2/2)

- A type is NOT a variable!
  - what are the differences between a type and a variable?
- The following is a definition of a type, NOT a declaration of a variable
  - A type needs to be defined before we can declare variable of that type
  - No memory is allocated to a type

```
typedef struct {
   int acctNum;
   float balance;
} account_t;
```

4.2 Structure Variables

- Declaration
  - The syntax is similar to declaring ordinary variables.

```
typedef struct {
   int stuNum;
   float score;
   char grade;
} result_t;
```

Before function prototypes
(but after preprocessor directives)

```
result_t result1, result2;
```

Inside any function

4.3 Initializing Structure Variables

- The syntax is like array initialization
- Examples:

```
typedef struct {
   int day, month, year;
} date_t;

typedef struct {
   int cardNum;
   date_t expiryDate;
} card_t;

card_t card1 = {888888, {31, 12, 2020}};
```

```
typedef struct {
   int stuNum;
   float score;
   char grade;
} result_t;

result_t result1 = { 123321, 93.5, 'A' };
```

4.4 Accessing Members of a Structure Variable

- Use the dot (.) operator

```
result_t result2;

result2.stuNum = 456654;
result2.score = 62.0;
result2.grade = 'D';
```

```
card_t card2 = { 666666, {30, 6} };

card2.expiryDate.year = 2021;
```

4.5 Example: Initializing and Accessing

StructureEg1.c

```
#include <stdio.h>

typedef struct  {
    int stuNum;
    float score;
    char grade;
} result_t;

int main(void) {
    result_t result1 = { 123321, 93.5, 'A' },
                result2;

    result2.stuNum = 456654;
    result2.score = 62.0;
    result2.grade = 'D';

    printf("result1: stuNum = %d; score = %.1f; grade = %c\n",
           result1.stuNum, result1.score, result1.grade);
    printf("result2: stuNum = %d; score = %.1f; grade = %c\n",
           result2.stuNum, result2.score, result2.grade);
    return 0;
}
```

```
result1: stuNum = 123321; score = 93.5; grade = A
result2: stuNum = 456654; score = 62.0; grade = D
```

Type definition

Initialization

Accessing members

4.6 Reading a Structure Member

- The structure members are read in individually the same way as we do for ordinary variables
- Example:

```
result_t result1;

printf("Enter student number, score and grade: ");

scanf("%d %f %c", &result1.stuNum, &result1.score,
                  &result1.grade);
```

4.7 Assigning Structures

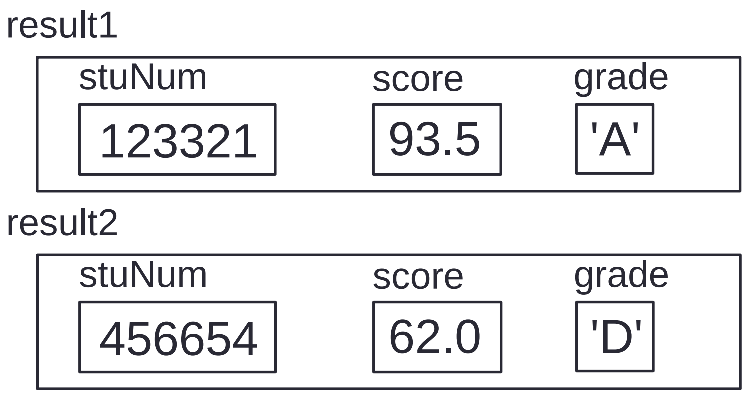
- We use the dot operator (.) to access individual member of a structure variable.
- If we use the structure variable's name, we are referring to the entire structure.
- Unlike arrays, we may do assignments with structures

```
result2 = result1;
```

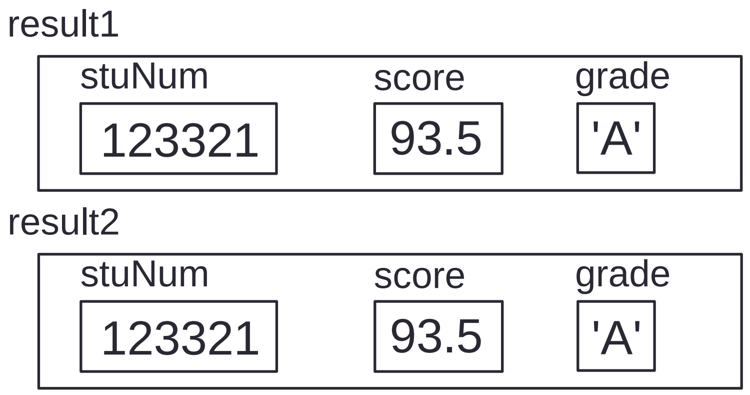
≡

```
result2.stuNum = result1.stuNum;
result2.score = result1.score;
result2.grade = result1.grade;
```

Before:



After:



4.8 Returning Structure from Function (1/3)

- Example:
  - Given this structure type result_t,

```
typedef struct {
      int max;
      float ave;
} result_t;
```

  - Define a function func() that returns a structure of this type:

```
result_t func( ... ) {
   ...
}
```

  - To call this function:

```
result_t result;

result = func( ... );
```

4.8 Returning Structure from Function (2/3)

StructureEg2.c

```
#include <stdio.h>

typedef struct {
    int max;
    float ave;
} result_t;

result_t max_and_average(int, int, int);

int main(void) {
    int num1, num2, num3;
    result_t result;

    printf("Enter 3 integers: ");
    scanf("%d %d %d", &num1, &num2, &num3);
    result = max_and_average(num1, num2, num3);

    printf("Maximum = %d\n", result.max);
    printf("Average = %.2f\n", result.ave);
    return 0;
}
...
```

returned structure is **copied** to *result*

max and average are printed

4.8 Returning Structure from Function (3/3)

StructureEg2.c

```
// Computes the maximum and average of 3 integers
result_t max_and_average(int n1, int n2, int n3) {
    result_t result;

    result.max = n1;
    if (n2 > result.max)
      result.max = n2;
    if (n3 > result.max)
      result.max = n3;

    result.ave = (n1+n2+n3)/3.0;

    return result;
}
```

the answers are stored in the structure variable *result*.

result is returned here

End of File