CYPRUS
INTERNATIONAL
UNIVERSITY

CYPRUS INTERNATIONAL UNIVERSITY
ENGINEERING FACULTY

Lecture 8

# Structures

CMPE223 / ISYE223
ALGORITHMS AND PROGRAMMING
Spring 2024-25

# Structures

- A **structure** is a **<u>collection of related data items</u>** grouped under a single name.
- Each data item in that structure is called a member.
- Unlike an array, a structure member can be any data type (even other structures).

# How to Create a Structure

- A structure can be created by using the `struct` keyword and declaring all of the members inside curly braces (`{}`).

structure tag

```
struct structureTypeName {
  members
};
```

- A structure definition does not reserve any space in memory.
- It creates a new data type.

# Examples

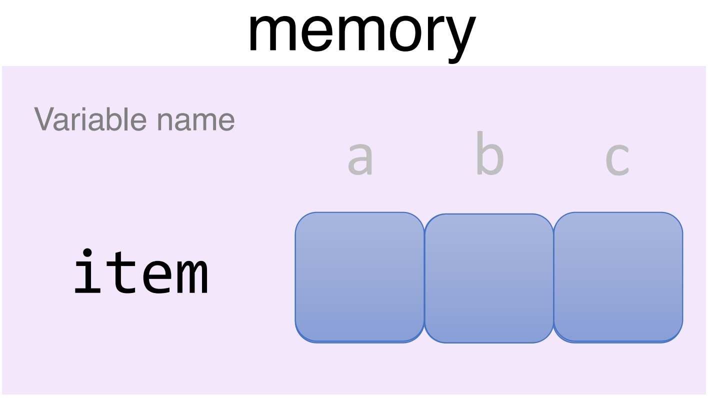
```
struct student{
  long num;
  char name[20];
  float cgpa;
  int age;
};

struct date{
     int day, month, year;
};
```

# Declaring Variables of Structures

```
struct st1{
      int a;
      float b;
      char c;
};

struct st1 item;
```

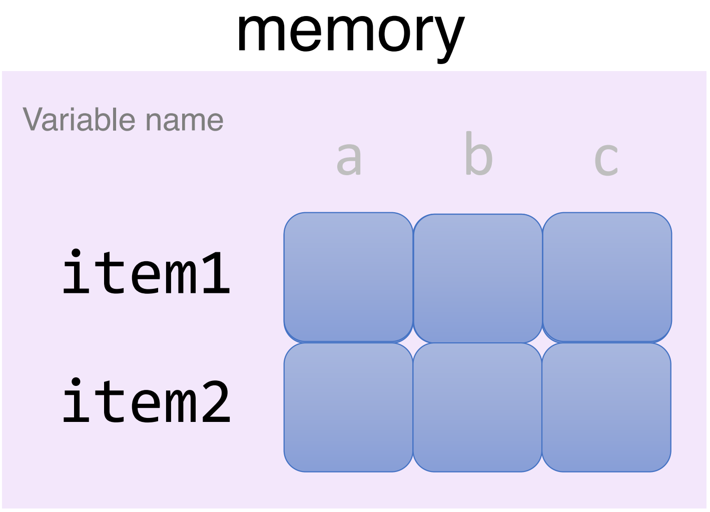
st1 data type is defined

A variable named item is created of st1 data type



# Declaring Variables of Structures (cont'd)

```
struct st1{
      int a;
      float b;
      char c;
}item1, item2;
```



Variables of a given structure type may also be declared by placing a comma-separated list of variable names between the struct's closing brace and terminating semicolon.

# Accessing Structure Members

- Structure members can be accessed with:

.

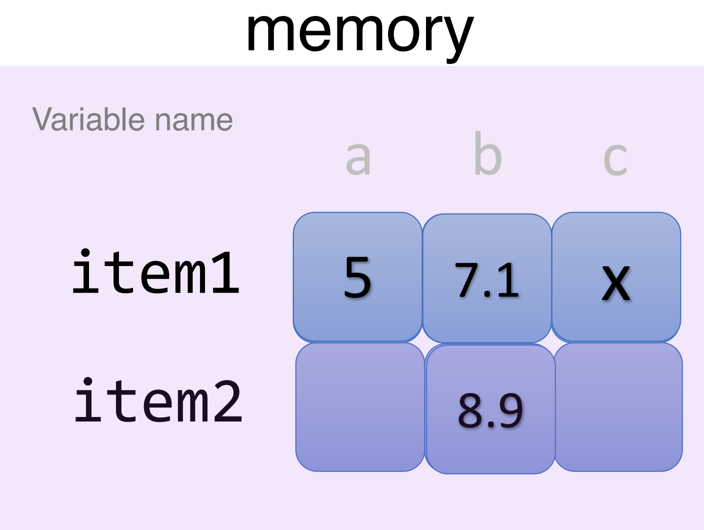
The structure member operator (dot operator)

- The structure member operator accesses a structure member via a structure variable name.

# Declaring & Initialising Struct Variables

```
struct st1{
     int a;
     float b;
     char c;
}item1;

struct st1 item2;

item1.a = 5;
item1.c = 'x';
item1.b = 7.1;
item2.b = 8.9;
```

Individual members of the structure are accessed using the dot operator.



memory

| Variable name | a | b | c |
|---|---|---|---|
| item1 | 5 | 7.1 | x |
| item2 |  | 8.9 |  |

# Two Different Ways to Initialise

```
struct date{
       int day, month, year;
} bday = {31, 12, 1988};
```

1a

Use a comma seperated list.

```
struct date bday = {31, 12, 1988};
```

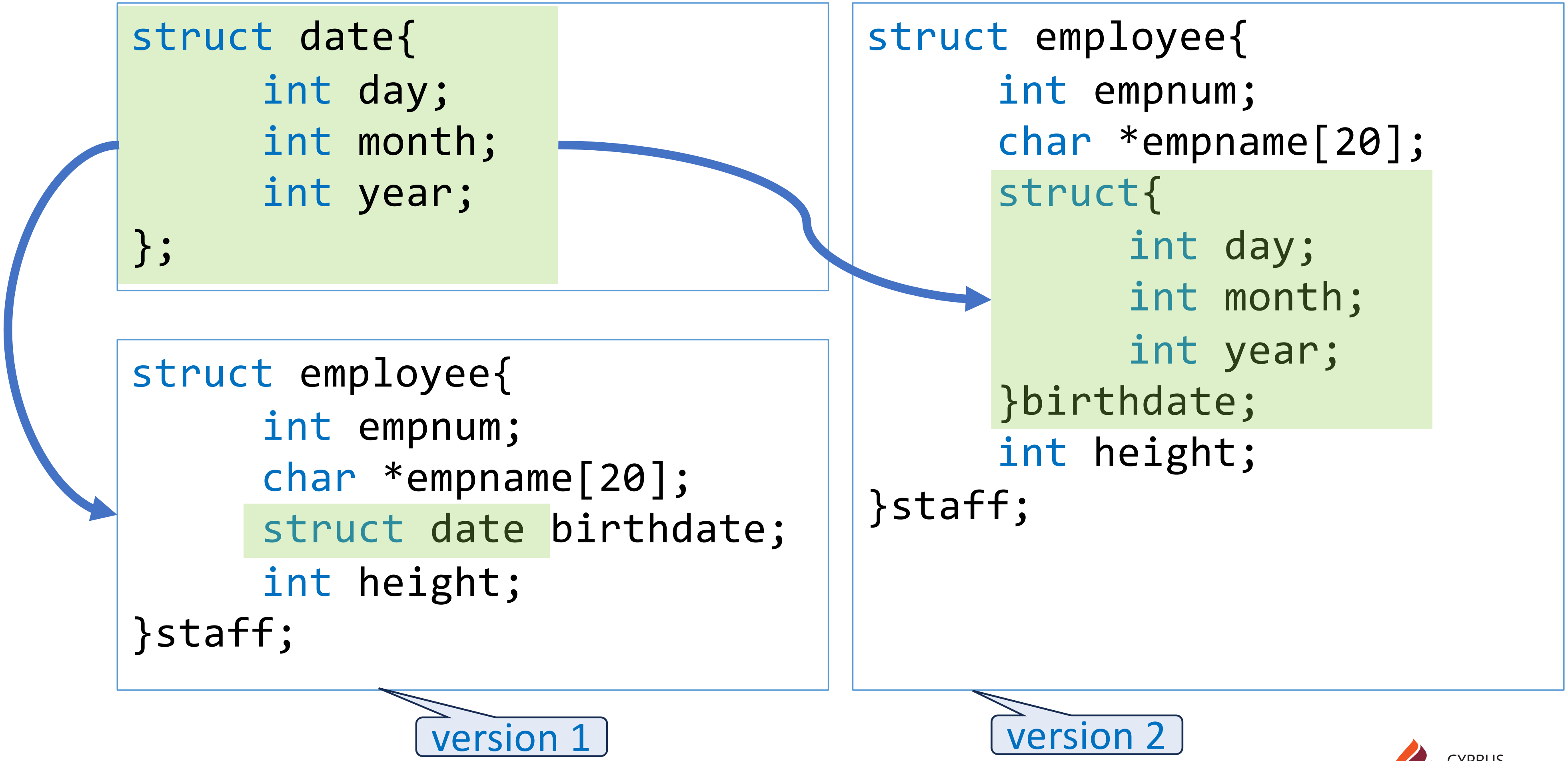
1b

```
struct date bday;
bday.day = 31;
bday.month = 12;
bday.year = 1988;
```

2

manual method

# Creating Nested Structures



```
struct date{
      int day;
      int month;
      int year;
};
```

```
struct employee{
      int empnum;
      char *empname[20];
      struct date birthdate;
      int height;
}staff;
```

version 1

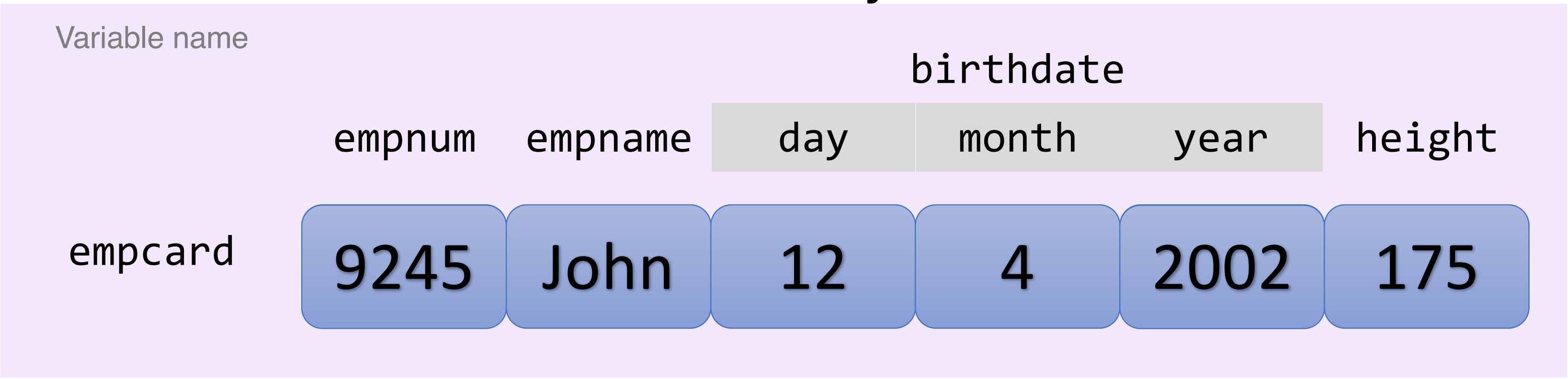
```
struct employee{
      int empnum;
      char *empname[20];
      struct{
            int day;
            int month;
            int year;
      }birthdate;
      int height;
}staff;
```

version 2

# Initialisation of Nested Structures

```
empcard.empnum = 9245;
empcard.empname = "John";          // if char *
//strcpy( empcard.empname, "John" ); // if char []
empcard.birthdate.day = 12;
empcard.birthdate.month = 4;
empcard.birthdate.year = 2002;
empcard.height = 175;
```

memory



| Variable name | | | birthdate | | | |
|---|---|---|---|---|---|---|
| | empnum | empname | day | month | year | height |
| empcard | 9245 | John | 12 | 4 | 2002 | 175 |

# Accessing Structure Members via a Pointer

- Structure members can be accessed with:

->

The structure pointer operator (arrow operator)

- A structure member can also be accessed via a pointer to the structure using the structure pointer operator — a minus (-) sign and a greater than (>) sign with no intervening spaces.

# Example

```
#include <iostream>
using namespace std;

struct date{
  int day;
  int month;
  int year;
};

struct employee{
  int empnum;
  //char *empname[20];
  const char *empname;
  struct date birthdate;
  int height;
};

int main(void){
  struct employee staff;
  struct employee *staffPtr = &staff;

  staff.empnum = 9245;
  //strcpy( staff.empname, "John" );
  staff.empname = "John";
  staff.birthdate.day = 12;
  staff.birthdate.month = 4;
  staff.birthdate.year = 2002;
  staff.height = 175;

  cout << staff.empname << endl;
  cout << staffPtr->empname << endl;
  cout << (*staffPtr).empname << endl;

  cout << staff.empnum << endl;
  cout << staff.birthdate.day << "/" << staff.birthdate.month << "/" << staff.birthdate.year << endl;
  cout << staff.height << endl;

}
```

# Comparing Structures

- Comparing Structures is Not allowed.
- Structure variables may not be compared using operators `==` or `!=`, because structure members may not be stored in consecutive bytes of memory.
- Sometimes there are "**holes**" in memory where a structure is stored. That's because computers store some data types only on certain memory boundaries (such as half-word, word or double-word boundaries which are machine-dependent).
- A word is a memory unit used to store data in a computer, usually four bytes (32-Bit) or eight bytes (64-Bit).

# Calculating The Size of The struct

```
#include <iostream>
using namespace std;

int main( void ){
   struct str1{
      int x;
      char c[10];
      float a;
      long b;
   };

   cout << "Manually adding member data types gives ";
   cout << sizeof(int) + (sizeof(char)*10) + sizeof(float) + sizeof(long);
   cout << " Bytes" << endl << "BUT" << endl ;

   cout << "Structure str1 is " << sizeof(str1) << " Bytes" << endl;

   return 0;
}
```

4
10
4
8

26 Bytes

Calculating the size manually

different

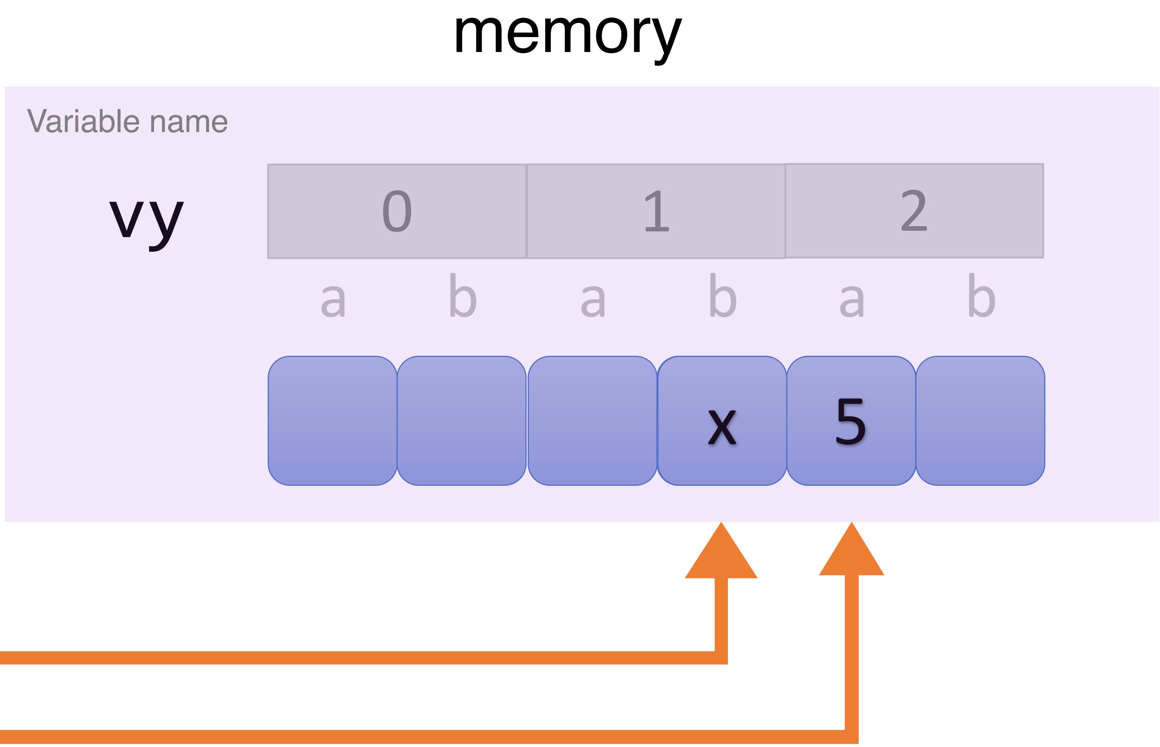
!

But str1 is 32 Bytes

# Arrays of Structs

Example:

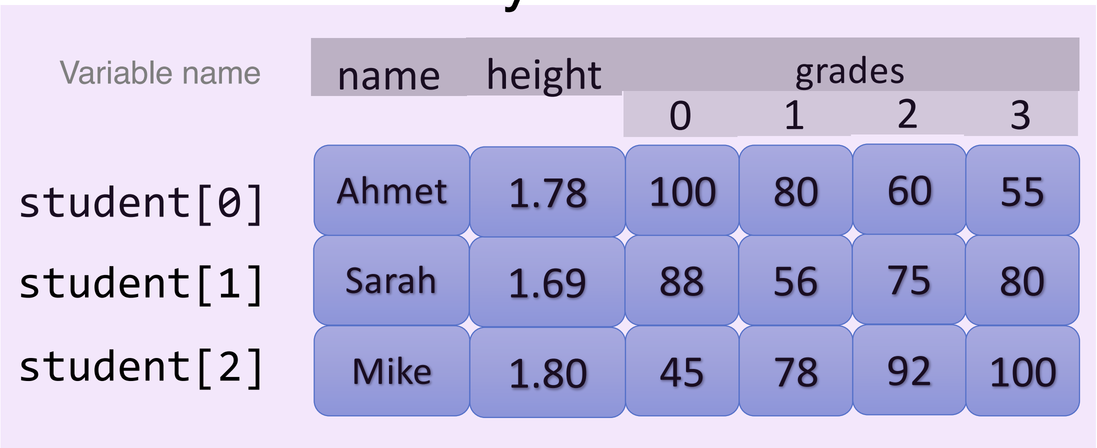
```
struct x {
   int a;
   char b;
} vy[3];

vy[1].b = 'x';
vy[2].a = 5;
```



# Example

```
struct stu{
        char name[10];
        float height;
        int grades[4];
} student[3] = {
    { "Ahmet", 1.78, {100, 80, 60, 55} },
    { "Sarah", 1.69,  {88, 56, 75, 80} },
    { "Mike",  1.80, {45, 78, 92, 100} }
  };
```

memory



| Variable name | name | height | grades | | | |
|---|---|---|---|---|---|---|
| | | | 0 | 1 | 2 | 3 |
| student[0] | Ahmet | 1.78 | 100 | 80 | 60 | 55 |
| student[1] | Sarah | 1.69 | 88 | 56 | 75 | 80 |
| student[2] | Mike | 1.80 | 45 | 78 | 92 | 100 |

# References & Links

- A Book on C, Fourth Edition, Al Kelley and Ira Pohl, Addison Wesley, 1999.
- C How to Program, Ninth Edition, Deitel & Deitel, Prentice Hall