UNIVERSITI TEKNIKAL MALAYSIA MELAKA

OPENCOURSEWARE

Advanced Programming (BETC 1353)

Week 5 : Structures, Unions, Bit Manipulations and Enumerations

Norfadzlia binti Mohd Yusof
norfadzlia@utem.edu.my
Ika Dewi binti Saiful Bahri
ikadewi@utem.edu.my

Learning Outcomes

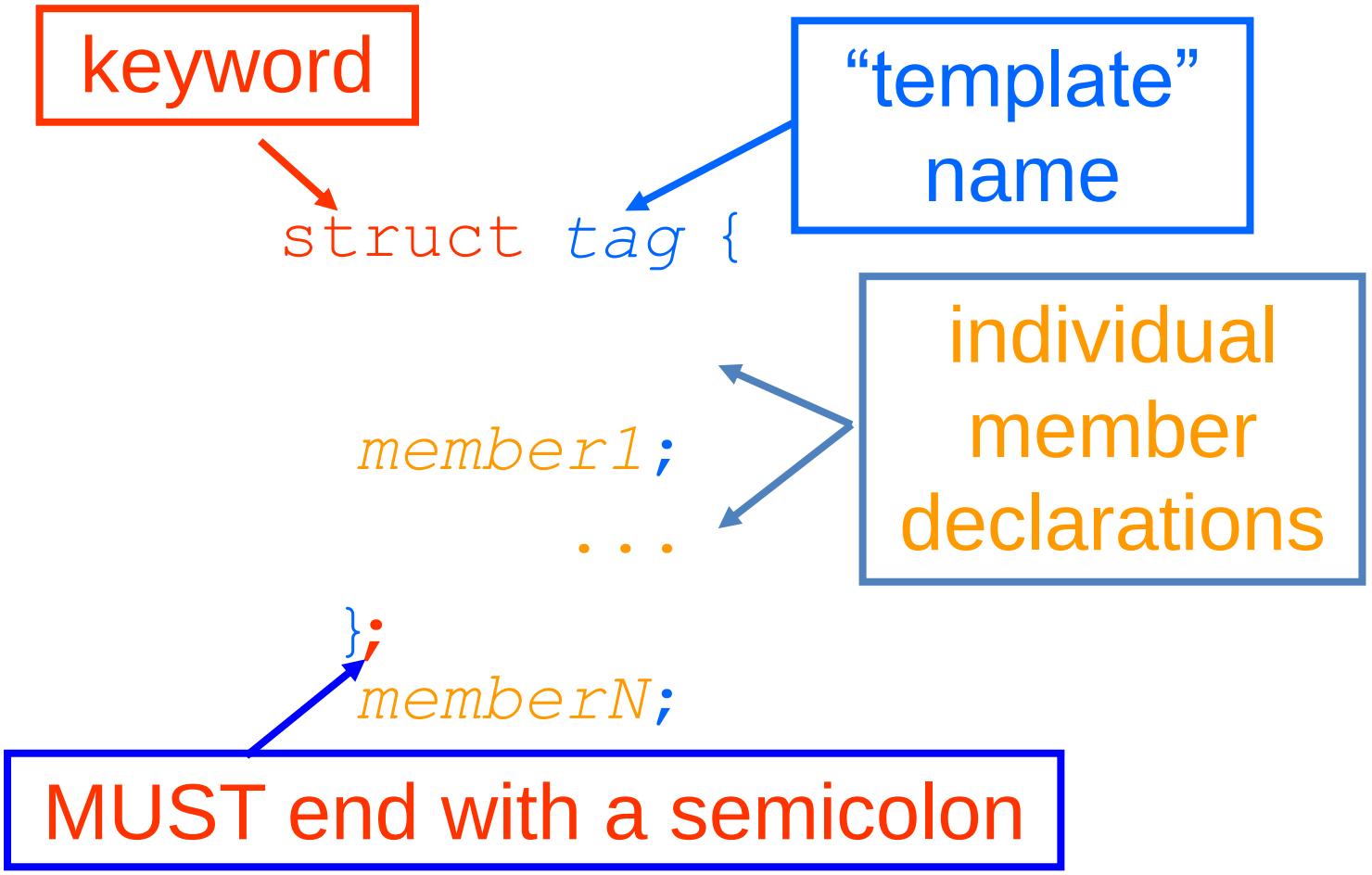
At the end of this lecture, you should be able to:

- Define and initialize structures
- Access structure members
- Combine structures and functions
- Define typedef

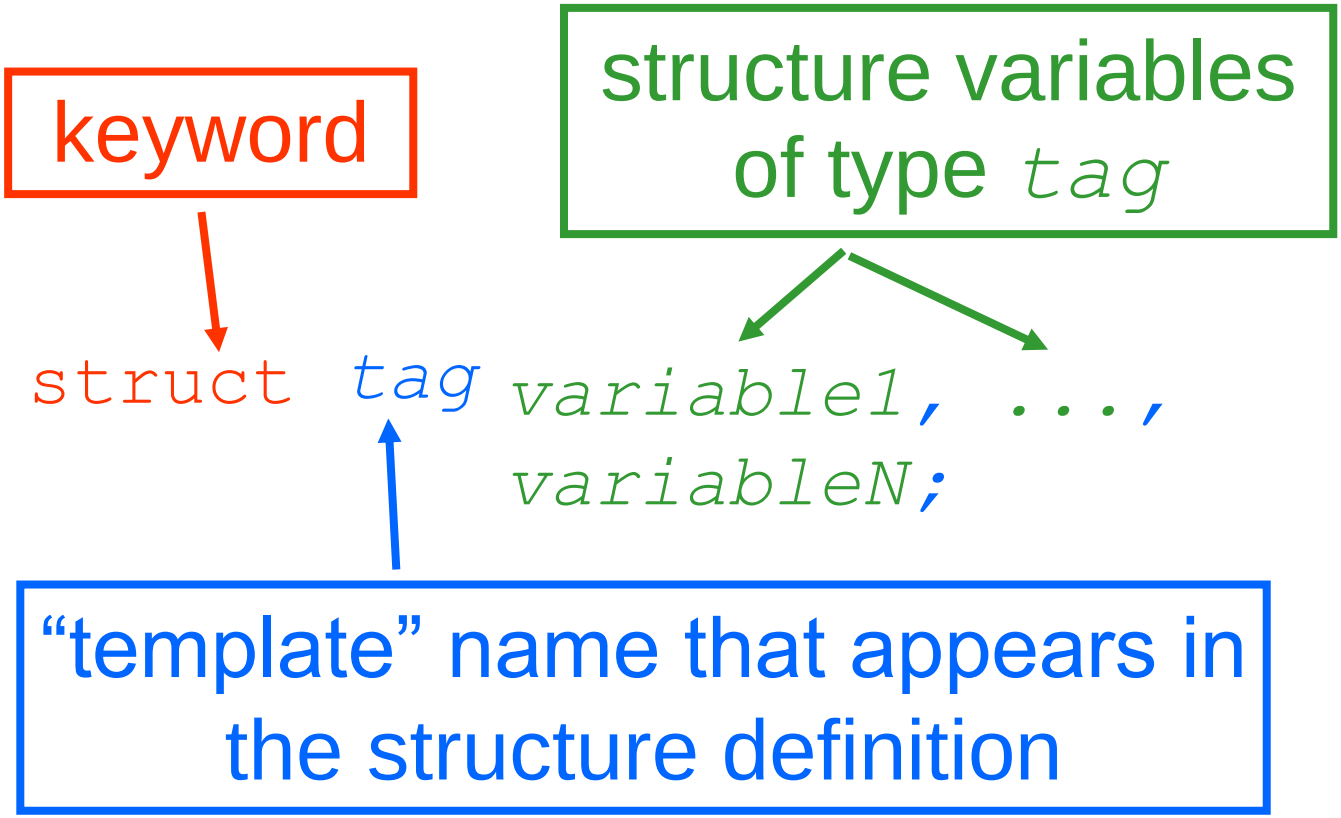
Structures

- What is a structure?
  - A collection of different data types under one name
  - Commonly used to define records to be stored in files
- Steps of using a structure
  - Define a “template”
  - Declare variables for the “template”

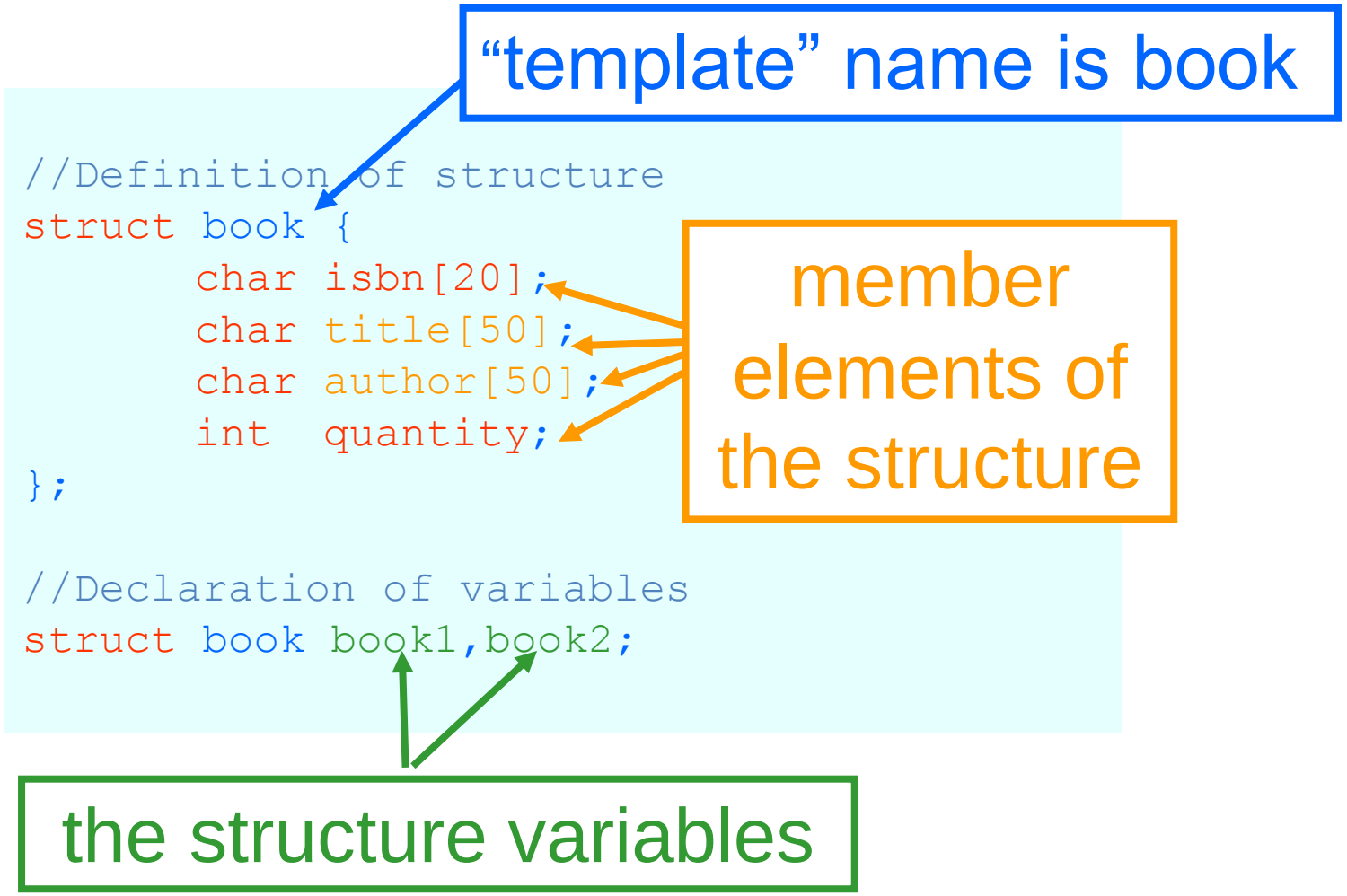
Defining a Structure



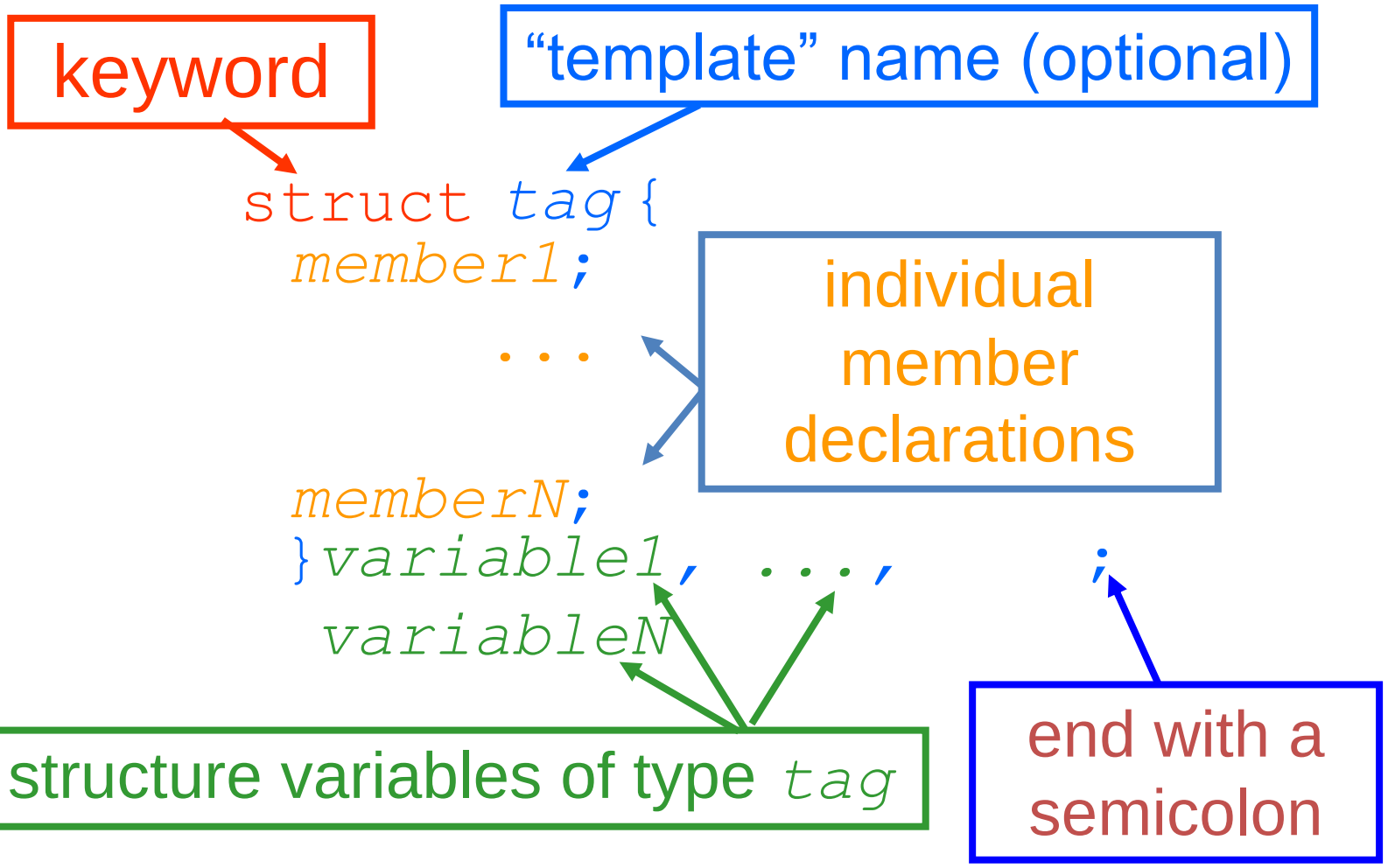
Declaring Variables



Example (I)



Combining Structure Declaration and Variables Definition



Example (II)

```
struct book{
        char isbn[20];
        char title[50];
        char author[50];
        int  quantity;
} book1, book2;
```

No tag

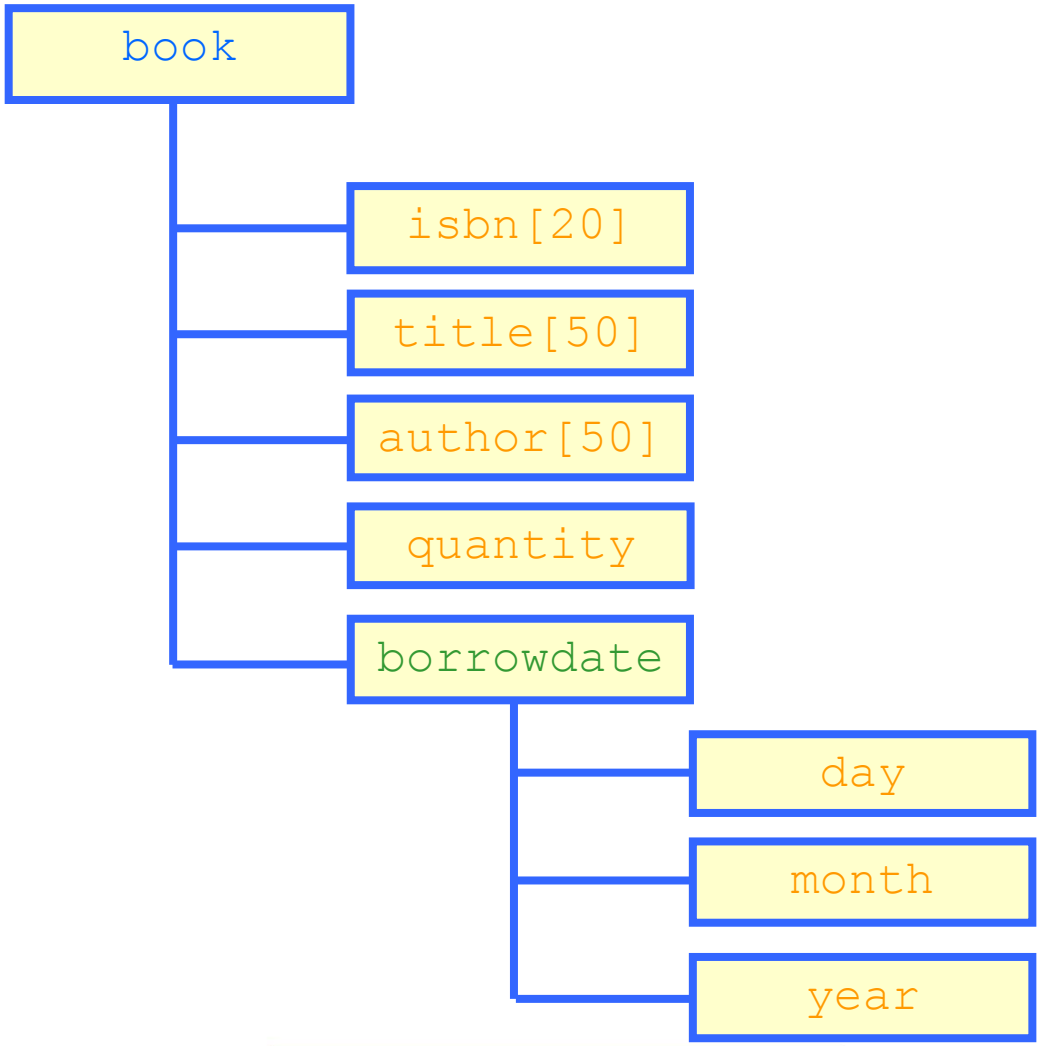
```
struct {
        char isbn[20];
        char title[50];
        char author[50];
        int  quantity;
} book1, book2;
```

Example (II)

- Advantages of using tags: Can reuse the "template" again

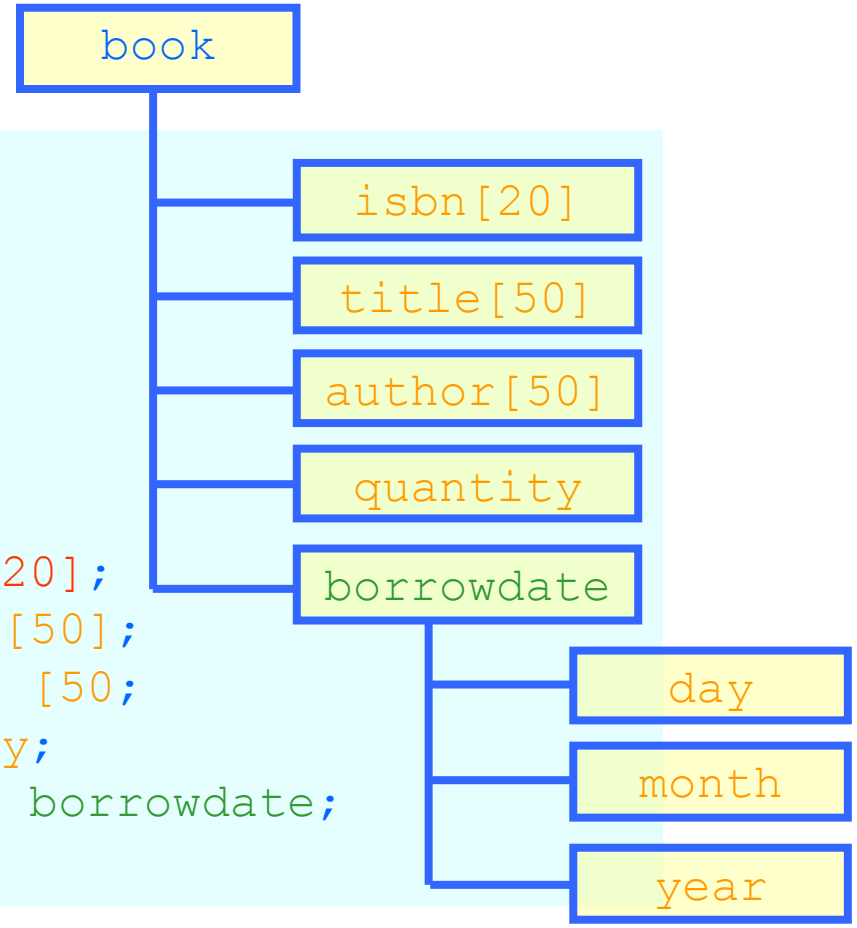
```
struct book{
        char isbn[20];
        char title[50];
        char author[50];
        int  quantity;
} book1, book2;

struct book book3;
```

Structures as Members



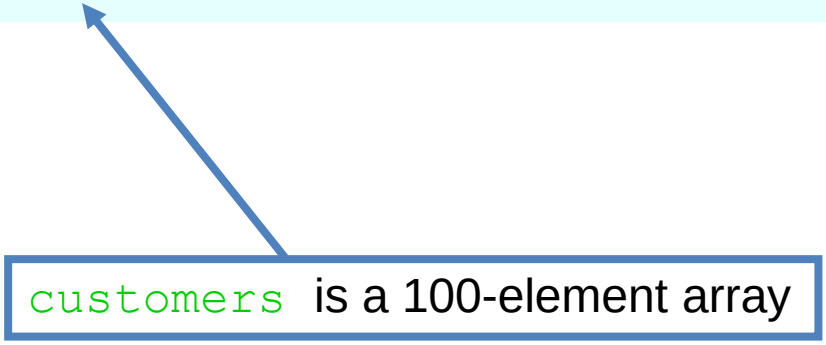
Structures as Members

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

struct book{
        char isbn [20];
        char title [50];
        char author [50;
        int quantity;
        struct date borrowdate;
} book1, book2;
```



Array of Structures

```
struct book{
        char isbn [20];
        char title [50];
        char author [50;
        int quantity;
        struct date borrowdate;
} books[100];
```



Initializing Structures

```
struct tag variable = {value1, ..., valueN};
```

```
struct book{
        char isbn[20];
        char title[50];
        char author[50;
        int quantity;
        struct date borrowdate;
}
```

must be in proper order as in structure definition

```
struct book book1 =
{123456, "Fundamental Programming", "Aki Ross", 5,
 11, 3, 2015};
```

isbn — title[50] — author — quantity — borrowdate

Initializing Structures (Array)

- For the case of initializing array of structures

```
struct book books[] = {
    {123456, "Learn C++", "Aki Ross", 5, 13, 3, 2015},
    {567891, "Learn JAVA", "Ryan Agner", 3, 25, 2, 2015}
    };
```

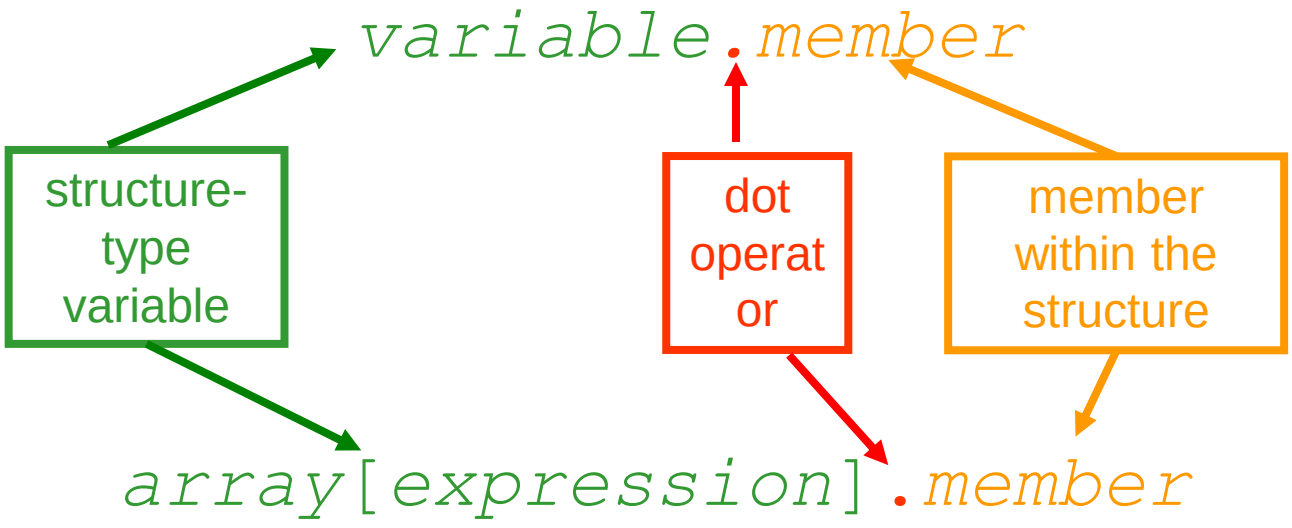
books[0]

books[1]

Valid Operators

- (=) to assign a structure to a structure of the same type
- (&) to get the address of a structure
- (.) to access the members of a structure
- (sizeof) to determine the size of a structure

Accessing Members of Structures




```
cout << "ISBN number=" << book1.isbn << endl;
cout << "Title= " << book1.title << endl;

int book_in;
book_in = book[1].quantity - 1;
```

Accessing Members of Structures

variable.member.submember

member of the embedded structure

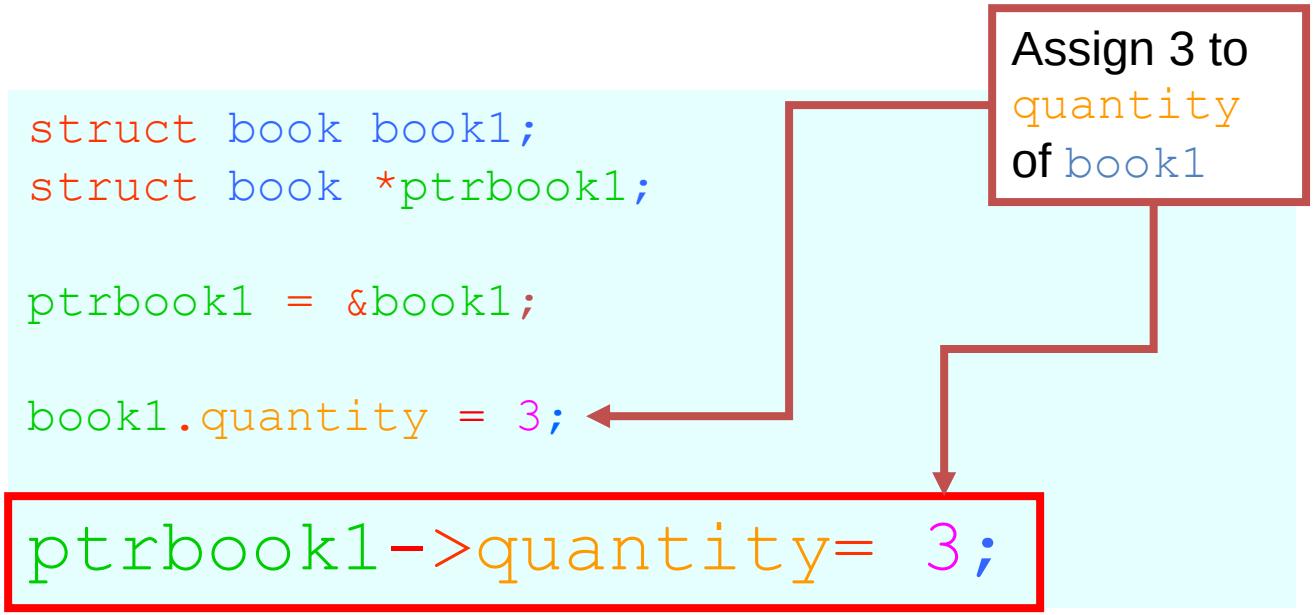
variable.member[expression]

structure member of type array

```
cout << "Year = " << book1.borrowdate.year << endl;

cout << book[1].title[0]) << endl;
```

Accessing Members of Pointers to Structure

- For the case of pointers to structure, usually member selection operator (->) is used to access the members



Accessing Members of Pointers to Structure

- Another alternative way is by using dereference operator (*) on the pointer
- After dereference, members can be accessed using dot operator (.)

```
(*ptrbook1).quantity = 3;
```

(*ptrbook1) ⇔ book1

Note:

(*ptrbook1).quantity ≠ *ptrbook1.quantity

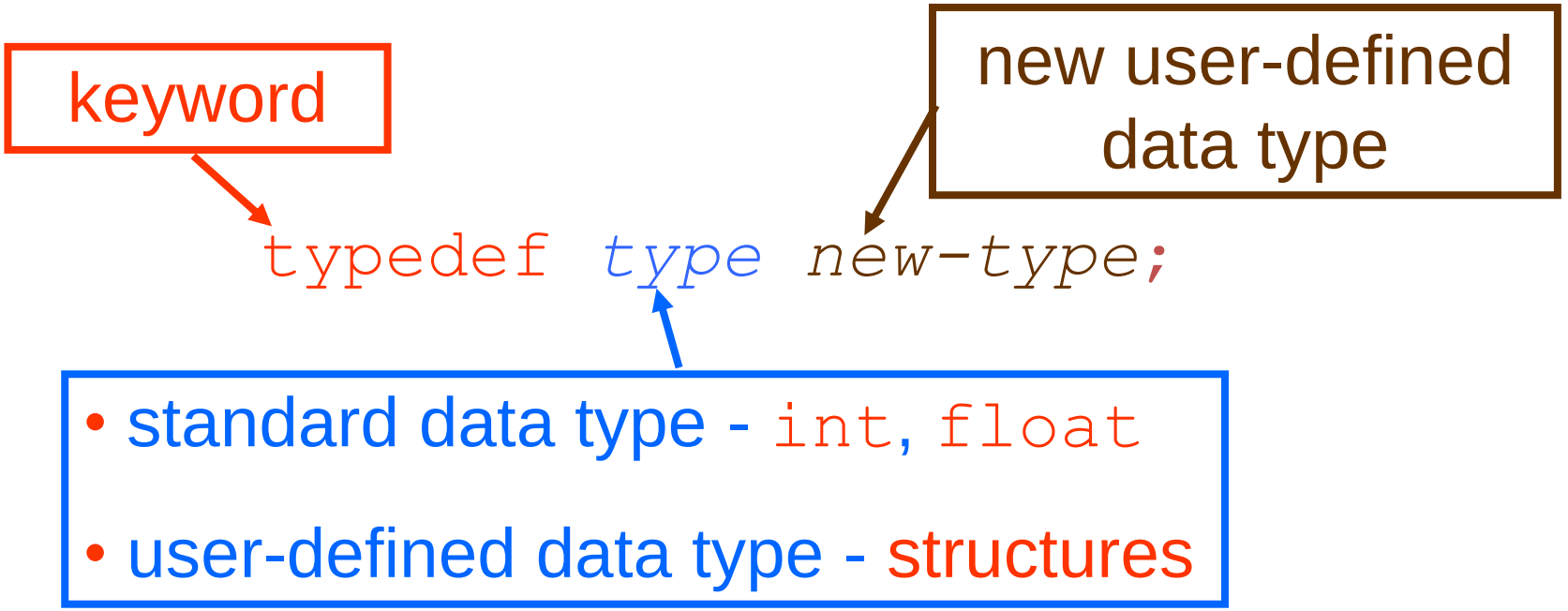
Accessing Pointer Members

```
struct test {
        int *val1;
        int *val2;
};

struct test testvar;
int a = 10;

testvar.val1 = &a;
cout << "Address is" << testvar.val1 << endl;
cout << "Value is" << *testvar.val1 << endl;
```

Shows the address of the pointer member `*val1`

Shows the value of the pointer member `*val1`, same as `*(testvar.val1)`

User-Defined Data Type




```
Typedef float Volume;

Volume box1, box2 ;
float box1, box2;
```

`Volume` is a user-defined data type equivalent to type `float`

Both are equivalent

User-Defined Data Type (`struct`)

"Template" name

```
typedef struct type new-type;
```

```
typedef struct {
    member1;
    ...
    memberN;
} new-type;
```

User-Defined Data Type (`struct`)

```
typedef struct {
        char isbn[20];
        char title[50];
        char author[50;
        int quantity;
} Record;

Record book1, book2;
```

User-Defined Data Type (`struct`)

```
typedef struct {
        int month;
        int day;
        int year;
} Date;

typedef struct {
        char isbn[20];
        char title[50];
        char author[50;
        int quantity;
        Date borrowdate;
} Record;

Record books[100];
```

Using Structures with Functions

- Passing individual structure members
  - pass by value
- Passing an entire structure
  - pass by value
- Passing a pointer to a structure
  - pass by reference

Passing Individual Structures Members

```
#include <iostream>
void printDate(int day, int month, int year);

typedef struct {
    int day;
    int month;
    int year;
} Date;

int main()
{
    Date returndate = {3, 12, 2015};

    printDate(returndate.day, returndate.month,
    returndate.year);

    return 0;
}
```

```
void printDate(int day, int
month, int year)
{
    cout << " Day = " << day;
    cout << "Month = " << month;
    cout << " Year = " << year <<
    endl;
}
```

Passing an Entire Structure

```
#include <iostream>
Using namespace std;
void printDate(Date);

typedef struct {
    int day;
    int month;
    int year;
} Date;

int main()
{
    Date borrowdate = {13, 3, 2015};

    printDate(borrowdate);

    return 0;
}
```

```
void printDate(Date returndate)
{
    cout << " Day = " <<
    returndate.day;
    cout << "Month = " <<
    returndate.month;
    cout << " Year = " <<
    returndate.year
         << endl;
}
```

Passing a Pointer to a Structure

```
#include <iostream>
void printDate(Date *);

typedef struct {
    int day;
    int month;
    int year;
} Date;

int main()
{
    Date borrowdate = {13, 3, 2015};

    printDate(&borrowdate);

    return 0;
}
```

```
void printDate(Date *brwPtr)
{
    cout << " Day = " <<
    brwPtr->day;
    cout << "Month = " <<
    brwPtr->month;
    cout << " Year = " <<
    brwPtr->year
         << endl;
}
```

Example

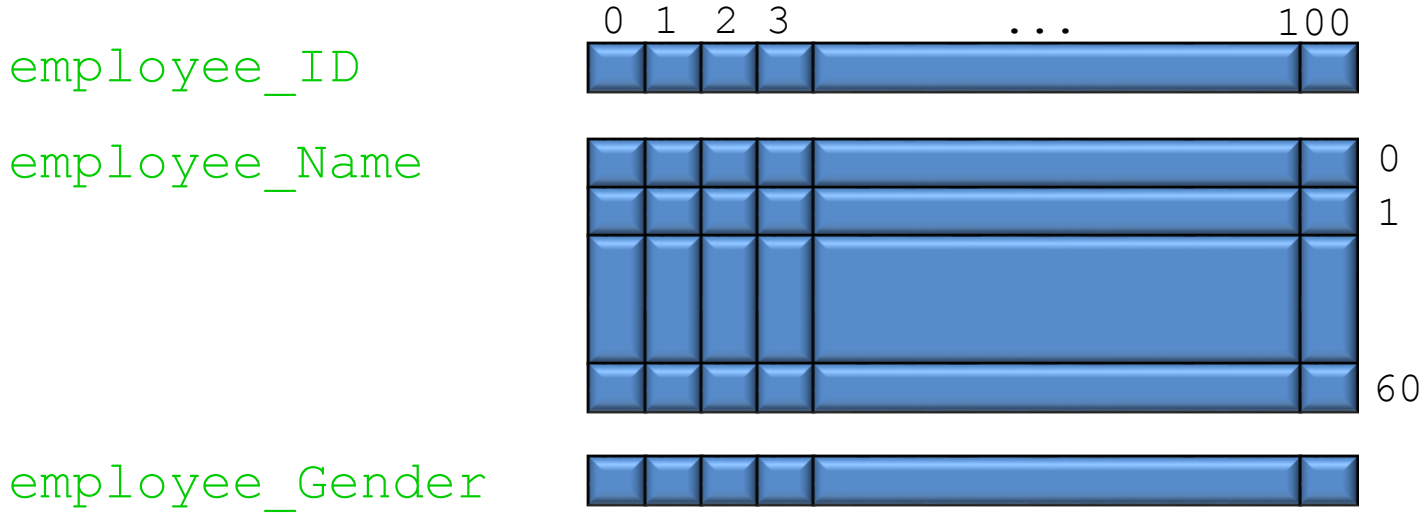
- Imagine a scenario where you're asked to write a program that stores records of 100 employees
- Each records contains
  - Staff ID (`int`)
  - Staff Name (`char[60]`)
  - Gender (`char`)

Storage Method - Without Structure

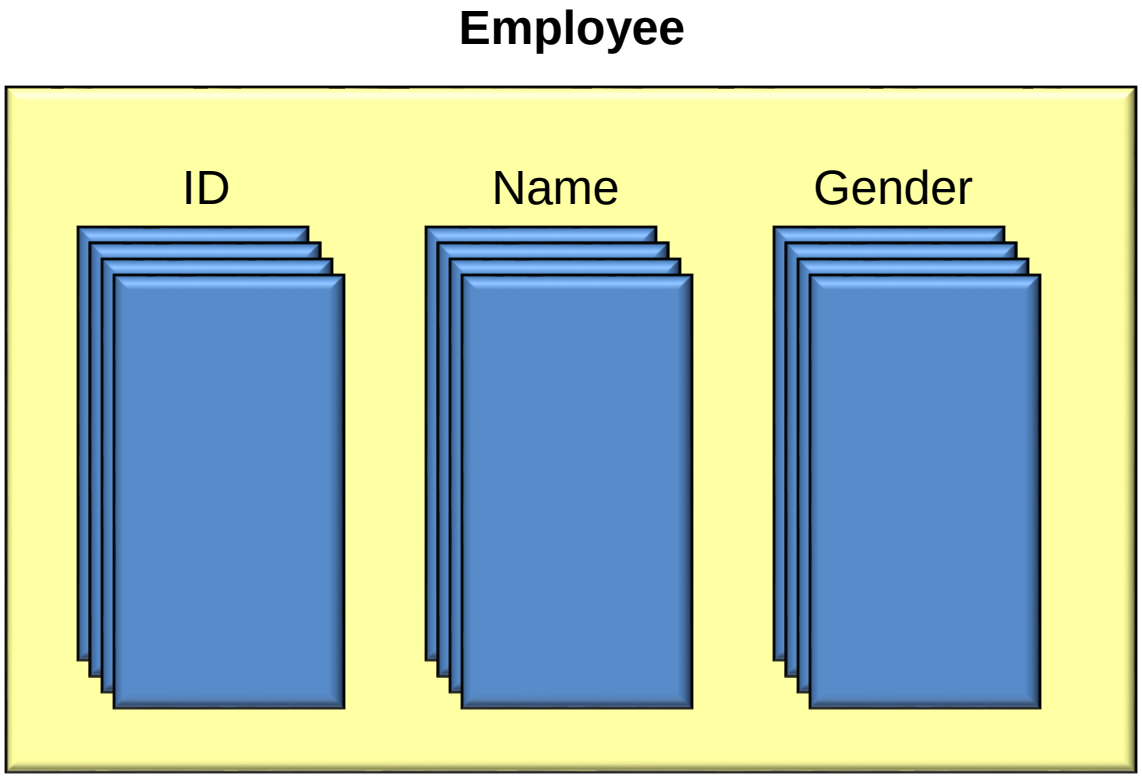
```
#define MAXRECORDS 100

int employee_ID[MAXRECORDS];
char employee_Name[MAXRECORDS][60];
char employee_Gender[MAXRECORDS];
```

The use of a 2D array.

Requires special attention and is much more difficult to handle.



Storage Method – Without Structure

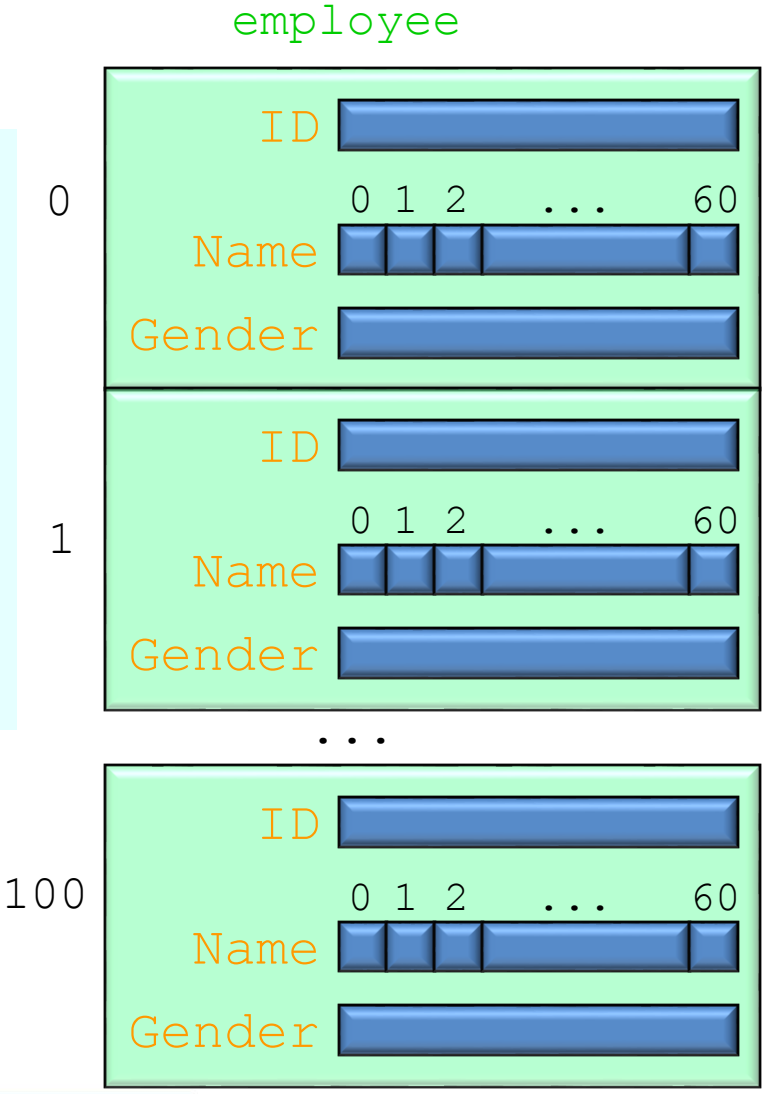


Uses 3 different array to store each item in the record.

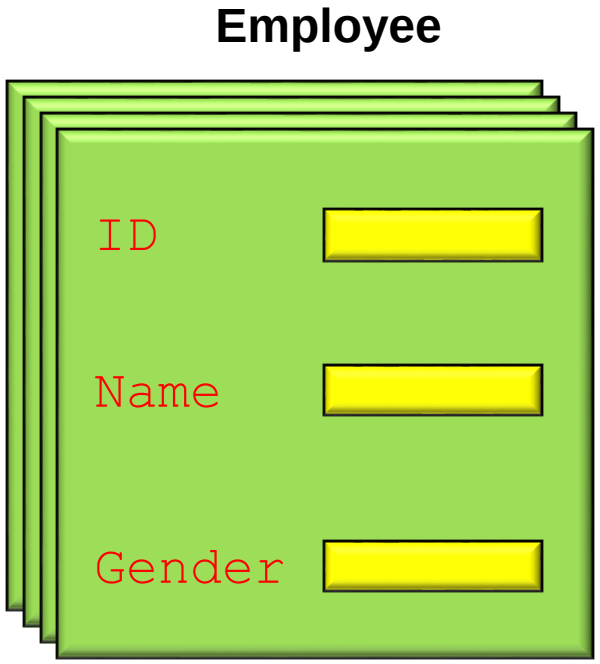
Not intuitive. Imagine using one file to store IDs, one to store names and another to store genders.

Storage Method – With Structure

```
#define MAXRECORDS 100

typedef struct {
    int ID;
    char Name[80];
    char Gender;
} Record;

Record employee[MAXRECORDS];
```



Storage Method – With Structure



Uses an array of structure. The data is stored as the member of the structure.

Simple and easy. Very intuitive, as it is similar to how we store hardcopy records using file.

Access Method - Without Structure

Function to print the info of a employee:

```
void printEmployee(int id, char *name, char gender){
      cout << "Employee Id:" << id << endl;
      cout << "Name:" << name << endl;
      cout << "Gender:" << gender << endl;
}
```

Needed to state all the data in the function individually.
Tedious and error-prone.

Statements to print all the employees using `printEmployee`:

```
int i;
for (i=0; i < MAXRECORDS; i++)
      printEmployee(employee_ID[i],
      &employee_Name[i][0],employee_Gender[i]);
```

Needed extra attention due to the use of 2D array.

Access Method – With Structure

Function to print the info of an employee:

```
void printEmployee(Record employee) {
      cout << "Employee Id:" << employee.ID << endl;
      cout << "Name:" << employee.Name << endl;
      cout << "Gender:" << employee.Gender << endl;
}
```

Statements to print all the employees using `printEmployee`:

```
int i;
for (i=0; i < MAXRECORDS; i++)
      printEmployee(employee[i]);
```

The whole structure can be passed into the function directly.
Simple and easy

Self-Review Questions

Question 1

Determine whether the following statements are valid or not?

a)

```
struct tax
   char names;
   float bills;
```

b)

```
struct class {
   char student;
   int number_of_student;
};

struct math{
   float marks;
   struct class record;
}test, final;
```

Self-Review Questions

Answer:

a)

```
struct tax {//complete the code here
   char names;
   float bills;
}; //complete the code here
```

Answer: Invalid

b)

```
struct class {
   char student;
   int number_of_student;
};

struct math{
   float marks;
   struct class record;
}test, final;
```

Answer: Valid

Self-Review Questions

Question 2

State the different between this two structure

```
struct product sale;
struct product *sale;
```

Self-Review Questions

Answer:

```
struct product sale;
```

Answer: normal variable

```
struct product *sale;
```

Answer: pointer variable

Self-Review Questions

Question 3

Complete the program below to display the result as shown.

```
#include <stdio.h>
#include <string.h>

typedef struct  BETC1353
{
  char student [60];
  float mark;
} record;

int main()
{
  record test;
  strcpy(test.student, "Ramlee");
  test.mark=90.3;
  return 0;
}
```

Result

Name is: Ramlee
Mark is: 90.30

Self-Review Questions

Answer:

```
#include <stdio.h>
#include <string.h>

typedef struct  BETC1353
{
  char student [60];
  float mark;
} record;

int main()
{
  record test;
  strcpy(test.student, "Ramlee");
  test.mark=90.3;
  printf("Name is: %s \n",test.student);
  printf("Mark is: %f\n", test.mark);
 return 0;
}
```

Self-Review Questions

- You are asked to write a program to stores the information of one patient in a Clinic Malaya
- The records contains:

  a. Patient Name : Aina Mardiana

  b. Patient ID : 12345

  c. Patient Contact Number: 0123456789

Self-Review Questions

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

struct patient {
         char name[80];
         int ID;
         int number;
};

int main ()
{
     struct patient index;
     index.name = "Aina Mardiana";
     index.ID = 12345;
     index.number = 0123456789;

     cout<<"Name: "<<index.name<<'\n';
     cout<<" ID number: "<<index.ID<<'\n';
     cout<<" Telephone number: "<<index.number;
     return 0;
}
```