Programming and Algorithms

COMP1038.PGA

Week 7 – Lectures 1, 2, 3: Structures

Dr. Pushpendu Kar

Outline

- Introduction
- Structure Definitions
- Initializing Structures
- Accessing Members of Structures
- Using Structures with Functions
- Typedef
- Union

Introduction

- Variables of primitive data types can only hold one value of a particular data type
- Array can hold multiple values of same data type
- In real life, data of different types is organized together
- Such as, Address book , library book information, student information, etc.
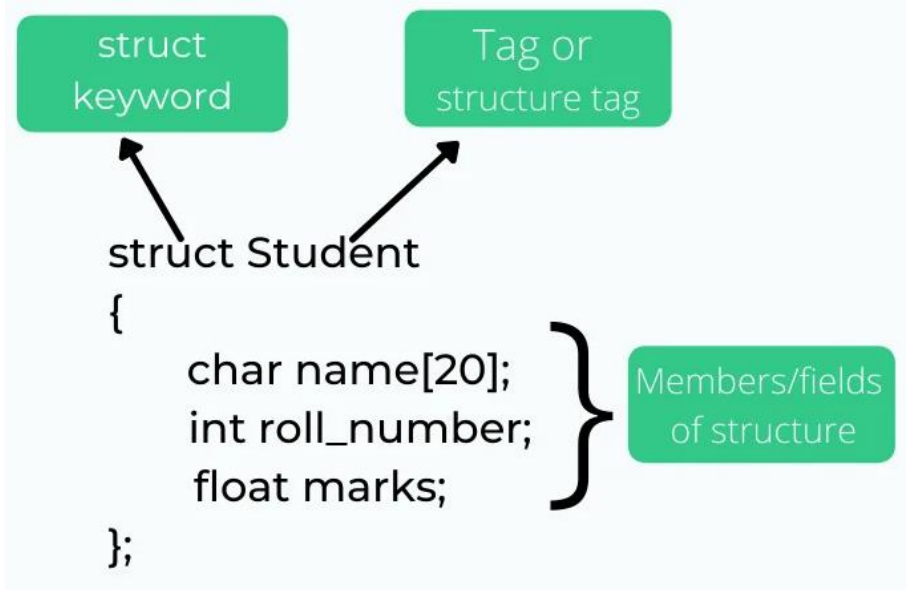- Structure and Union are solution for this.

Structures

- Collections of related variables (aggregates) under one name
- Can contain variables of different data types
- Commonly used to define records to be stored in files
- Combined with pointers, can create linked lists, stacks, queues, and trees

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Syntax Structure definition

struct <structure tag>

{

structure element 1 ;

structure element 2 ;

structure element 3 ;

……

} ;

Source: https://www.geeksforgeeks.org/

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Structure Definitions

- Example

```
struct Student {
        char name [20];
        char address [20];
        int rollno;
        char class[5];
        char year[20];
};
```

- struct introduces the definition for structure Student
- Student is the structure name and is used to declare variables of the structure type

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Structure Definitions cont...

- **Structure information**
  - A structure cannot contain an instance of itself
  - Can contain a member that is a pointer to the same structure type
  - A structure definition does not reserve space in memory
    - Instead creates a new data type used to declare structure variables
- **Declarations**
  - Declared like other variables:

```
struct Student oneStud, Stud[66], *SPtr;
```

- **Can use a comma separated list:**

```
struct Student {
        char name [20];
        char address [20];
}oneStud, Stud[66], *SPtr;
```

Declaration of Struct variable

- struct book
  {
    char name ;
    float price ;
    int pages ;
  } b1, b2, b3 ;

  Inline structure declaration

  struct book
  {
    char name ;
    float price ;
    int pages ;
  } ;
  struct book b1, b2, b3;

  General structure declaration

- struct
  {
    char name ;
    float price ;
    int pages ;
  } b1, b2, b3 ;

  Anonymous structure declaration

  N. B. If you declare structure variables through anonymous structure declaration then you can't declarer more structure variables later of the same structure type.

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Structure Definitions cont…

- Valid Operations
  - Assigning a structure to a structure of the same type
  - Taking the address (&) of a structure
  - Accessing the members of a structure
  - Using the *sizeof* operator to determine the size of a structure

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Initializing Structures

- Initialize structures
  - Example:

```
struct Student {
        char name [20];
        char address [20];
        int rollno;
        char class[5];
        char year[20];
        };
struct Student S1 = {"Peter", "Ningbo", 2543, "CS", "2021-2022"};

struct book
  {
    char name[10] ;
    float price ;
    int pages ;
  } ;
  struct book b1 = { "Basic", 200.50, 550 } ;
  struct book b2 = { "Physics", 300.48, 800 } ;
```

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Assignment statements

- Example:

  struct Student S2 = S1;

  Same type of structure variables can directly copy from one another using assignment operator

- Could also declare and initialize as follows:

  struct Student S2;
  S2.name = "Peter";
  S2.address = "Ningbo";
  S2.rollno = 2543;
  S2.class = "CS";
  s2.year="2021-2022";

  Member-by-member initialization

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Accessing Members of Structures

- Accessing structure members
  - Dot operator (.) used with structure variables
    ```
    struct Student S1;
    printf( "%s", S1.name );
    ```
  - Arrow operator (->) used with pointers to structure variables
    ```
    struct Student *SPtr = &S1;
    printf( "%s", SPtr->name );
    ```
  - SPtr->name is equivalent to
    ```
    ( *SPtr ).name;
    ```

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Memory allocation cont…

```
struct book
{
    char name;
    float price;
    int pages;
};
struct book b1 = {'B',130.00,550};
```

b1.name	b1.price	b1.pages
'B'	130.00	550
65518	65519	65523

Array of Structures

```c
#include <stdio.h>

void main( )
{
    struct book
    {
        char name;
        float price;
        int pages;
    };
    struct book b[3];
    int i;
    char c;

    for (i = 0; i < 3; i++)
    {
        printf ("Enter name, price and pages:  ");
        scanf ("%c %f %d", &b[i].name, &b[i].price, &b[i].pages) ;
        while ((c=getchar()) != '\n' && c != EOF);
    }
    printf ("\n%c %.1f %d\n\n", b[1].name, b[1].price, b[1].pages);
}
```

```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_1
Enter name, price and pages:  A 100.5 200
Enter name, price and pages:  B 110.8 250
Enter name, price and pages:  C 150.3 300

B 110.8 250
```

Array of structure
Format: structure variable name [size]
Like normal array index start from o

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

Additional Features of Structures

1. Structure and Assignment operator

- The values of a structure variable can be assigned to another structure variable of the same type using the assignment operator.
- No piecemeal copy is needed .

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Additional Features of Structures cont...

Structure and Assignment operator

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

void main( )
{
    struct employee
      {
          char name[10];
          int age;
          float salary;
      };
struct employee e1 = {"Peter", 30, 10000.00};
struct employee e2, e3;

strcpy (e2.name, e1.name);
e2.age = e1.age;
e2.salary = e1.salary;

e3 = e2;

printf("\n%s %d %.2f", e1.name, e1.age, e1.salary);
printf("\n%s %d %.2f", e2.name, e2.age, e2.salary);
printf("\n%s %d %.2f", e3.name, e3.age, e3.salary);
}
```

```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_2

Peter 30 10000.00
Peter 30 10000.00
Peter 30 10000.00
```

Member-by-member copy

Copying all the elements at one go

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Additional Features of Structures cont...

2. Nested structure: One structure can be nested within another structure. Using this facility complex data types can be created. The following program shows nested structures at work.

```c
#include <stdio.h>

void main( )
{
  struct address
   {
     char phone[15];
     char city[25];
     int pin;
   };

   struct emp
   {
     char name[25];
     struct address a;          ← Nested structure
   };

struct emp e = { "Peter", "1234567890", "Ningbo", 315100 };

printf("\nname = %s \nphone = %s", e.name, e.a.phone);
printf("\ncity = %s \npin = %d", e.a.city, e.a.pin);
}
```

```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_3

name = Peter
phone = 1234567890
city = Ningbo
pin = 315100
```

Look carefully how to access nested structure member

3. Structures With Functions

- **Passing structures to functions**
  - Pass entire structure
  - Or, pass individual members
  - Both pass call by value
- **To pass structure call-by-value**
  - Like an ordinary variable, a structure variable can also be passed to a function.
  - Or whole structure can be passed to a function
- **To pass structures call-by-reference**
  - Pass its address
  - Pass reference to it

Additional Features of Structures cont…

3. Passing individual structure elements

```c
#include <stdio.h>

void display(char*, char*, int);
void main( )
{
  struct book
   {
     char name[25];
     char author[25];
     int callno;
   };

  struct book b1 = {"C How to program","Paul Deitel",101};

  display(b1.name, b1.author, b1.callno);
}

void display (char *s, char *t, int n)
{
    printf("\n %s %s %d", s, t, n) ;
}
```

```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_4

C How to program Paul Deitel 101
```

Passing structure members

Called function

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Additional Features of Structures cont...

3. Passing entire structure variable at a time

```c
#include <stdio.h>

struct book
 {
     char name[25];
     char author[25];
     int callno;
 };

void display(struct book);

void main( )
{
  struct book b1 = {"C How to program","Paul Deitel",101};

  display(b1);
}

void display (struct book b)
{
     printf ("\n%s %s %d", b.name, b.author, b.callno );
}
```

```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_5

C How to program Paul Deitel 101
```

Passing the entire structure

Called function

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Structure and pointer

- The way we can have a pointer pointing to an int, or a pointer pointing to a char.
- Similarly we can have a pointer pointing to a structure.
- Such pointers are known as 'structure pointers'

The University of
Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Structure and pointer cont...

```c
#include <stdio.h>

void main ()
{
  struct book
  {
   char name[25];
   char author[25];
   int callno;
  };

  struct book b1 = {"C How to program","Paul Deitel",101};

  struct book *ptr;

  ptr = &b1;

  printf("\n%s %s %d", b1.name, b1.author, b1.callno);
  printf("\n%s %s %d", ptr->name, ptr->author, ptr->callno);
}
```

```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_6

C How to program Paul Deitel 101
C How to program Paul Deitel 101
```

Structure variable

Defining structure pointer

Creating structure pointer

Accessing structure members through structure pointer

Structure and pointer memory map

b1.name	b1.author	b1.callno
C How to program	Paul Deitel	101
65472	65497	65522

ptr

65472

65529

Call by reference and structure

Passing address of a structure variable

```c
#include <stdio.h>

struct book
{
char name[25];
char author[25];
int callno;
};

void display(struct book *);

void main()
{
  struct book b1 = {"C How to program","Paul Deitel",101};
  display(&b1);
}

void display (struct book *b)
{
  printf ("\n%s %s %d", b-> name, b->author, b->callno);
}
```

```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_7

C How to program Paul Deitel 101
```

Passing structure pointer

Called function

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Array of structure and pointer

```c
#include <stdio.h>

struct book
{ char name[25];
char author[25];
int callno;
};

void display(struct book *, int);
int input(struct book *);

void main()
{ int n;
  struct book b1[20];
  n= input (b1);
  display (b1,n);
}
```

Passing pointer of structure array

Called function

```c
void display(struct book *b , int n)
{
  int i;
  printf("\n\t\t Books Available ");
  printf("\n\t Sr. No \t Name \t Author \t Call No \n");

  for(i=0;i<n;i++)
  {
    printf("\n\t %d \t\t %s \t %s \t %d",i+1, (b+i)->name, (b+i)->author, (b+i)->callno);
  }
}
```

Input function

```c
int input(struct book *b)        ← Called function
{
  int i, n;
  char c;
  printf("\n\t\t Enter No of books ");
  scanf("%d",&n);

  for(i=0;i<n;i++)
  {
    printf("\n\t Sr. No %d ", i+1);
    printf("\n\t Name: ");

    while ((c=getchar()) != '\n' && c != EOF);
    gets((b+i)-> name);
    printf("\t Author Name: ");
    gets((b+i)-> author);
    printf("\t Call No: ");
    scanf("%d",&(b+i)->callno);
  }
  return n;
}
```

```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_8
                Enter No of books 2

        Sr. No 1
        Name: C How to program
        Author Name: Paul Deitel
        Call No: 101

        Sr. No 2
        Name: Let us C
        Author Name: YPK
        Call No: 102

                Books Available
        Sr. No          Name     Author          Call No

        1                       C How to program     Paul Deitel     101
        2                       Let us C         YPK     102
```

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Uses of Structures

- Data base management
- Changing the size of the cursor
- Clearing the contents of the screen
- Placing the cursor at an appropriate position on screen
- Drawing any graphics shape on the screen
- Receiving a key from the keyboard
- Checking the memory size of the computer
- Finding out the list of equipment attached to the computer
- Hiding a file from the directory
- Displaying the directory of a disk
- Sending the output to printer
- Interacting with the mouse

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Thank you!

The University of
Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

What would be the output of the following program?

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

void main()
{
    struct gospel
    { int num;
      char mess1[50];
      char mess2[50];
    }m;
  m.num = 1;
  strcpy(m.mess1, "If all that you have is hammer");
  strcpy(m.mess2, "Everything looks like a nail");
  printf("\n%u %u %u", &m.num, m.mess1, m.mess2);
}
```

```
[z2019024@CSLinux pga]$ ./structure8

1397932336 1397932340 1397932390
```

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

What would be the output of the following program?

```c
#include <stdio.h>

struct gospel {
int num ;
char mess1[50];
char mess2[50];
} m1 = {2, "If you are driven by success", "make sure that it is a quality drive"};

void main( )
{
    struct gospel m2, m3;
    m2 = m1;
    m3 = m2;
    printf ("\n%d %s %s", m1.num, m2.mess1, m3.mess2);
}
```

```
[z2019024@CSLinux pga]$ ./structure8

2 If you are driven by success make sure that it is a quality drive
```

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Point out the errors, if any, in the following programs:

```
1   #include <stdio.h>
2   #include <string.h>
3
4   void main( )
5   { struct employee
6     { char name[25];
7       int age;
8       float bs;
9     };
10    struct employee e;
11    strcpy (e.name, "Hacker");
12    age = 25;
13    printf ("\n%s %d", e.name, age);
14  }
```

1. At line 12, age will be e.age
2. At line 13, age will be e.age

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Point out the errors, if any, in the following program:

```
1    #include <stdio.h>
2
3    void main( )
4    {
5      struct
6      { char name[25];
7        char language[10];
8      };
9      struct employee e = {"Hacker", "C"};
10     printf ("\n%s %d", e.name, e.language);
11   }
```

1. At line 5, structure name is missing. It will be 'struct employee'
2. At line 10, 2nd format specifier of printf statement will be %s instead of %d

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Point out the errors, if any, in the following program:

```
1    #include <stdio.h>
2
3    struct virus
4    { char signature[25];
5      char status[20];
6      int size;
7    } v[2] = {"Yankee Doodle", "Deadly", 1813, "Dark Avenger", "Killer", 1795};
8
9    void main( )
10   { int i;
11     for (i = 0 ; i <=1 ; i++)
12       printf ("\n%s %s", v.signature, v.status);
13   }
```

1. At line 12, array indexes of 'v.signature' and 'v.status' are missing. They will be 'v[i].signature' and 'v[i].status'

The University of **Nottingham**

UNITED KINGDOM · CHINA · MALAYSIA

typedef

- typedef
  - Creates aliases for previously defined data types
  - Use typedef to create shorter type names
- Typedef allows us to associate a name with a structure (or other data type).

typedef

Put typedef at the start of your program.

```c
#include <stdio.h>

typedef struct line {
    int x1, y1;
    int x2, y2;
} LINE;

void main()
{
    LINE line1;
    line1.x1 = 12;
    line1.y1 = 14;
    line1.x2 = 50;
    line1.y2 = 52;
    printf("%d, %d, %d, %d", line1.x1,line1.y1,line1.x2,line1.y2);
}
```

line1 is now a structure of line type

```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_14
12, 14, 50, 52
```

The University of **Nottingham**

UNITED KINGDOM · CHINA · MALAYSIA

Typedef Example

```
typedef struct
{ char *first;
  char *last;
  char SSN[9];
  float gpa;
  char **classes;
} student;


student stud_a, stud[20] ,*sptr;
```

student is now 'struct student'

Typedef Example

```
struct employee
{ char name[25];
  int age;
  float bs;
}
typedef empoyee emp;
```

emp is now 'struct employee'

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Union

- Like structures, but every member occupies the same region of memory!
  - Structures: members are "and"ed together
  - Unions: members are "xor"ed together

```
union VALUE
{ float f;
  int i;
  char *s;
};
```

The University of
Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Union

- Storage
  - size of union is the size of its largest member
  - avoid unions with widely varying member sizes;
  - for the larger data types, consider using pointers instead
- Initialization
  - Union may only be initialized to a value appropriate for the type of its first member

```c
#include <stdio.h>

union Test
{
    int x;
    char c;
};

void main()
{
    union Test t = {1};

    printf("%d, %c", t.x, t.c);
}
```
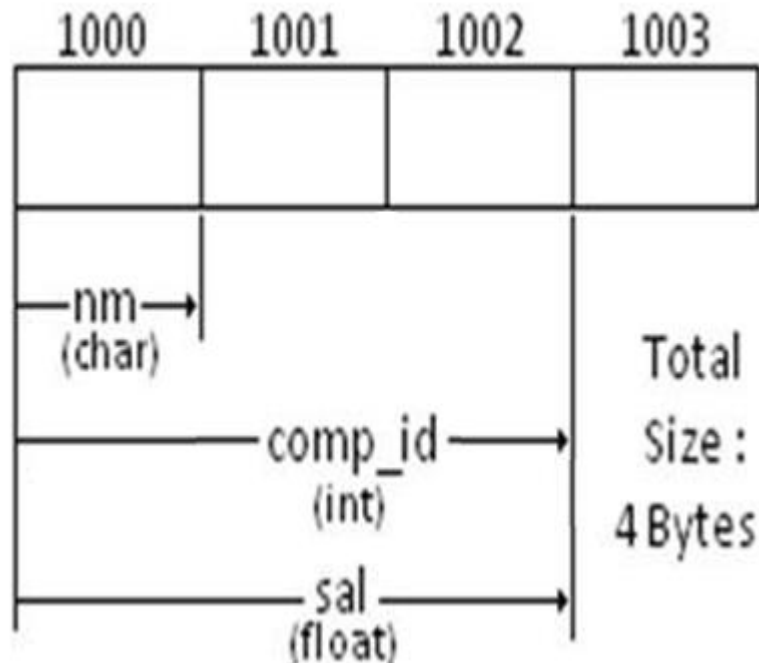
```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_15
1, □□
```

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Memory allocation and accessing elements


```
union techno
{ char nm;
  int comp_id;
  float sal;
}tch;
```

Accessing element:
```
tch.nm
tch.comp_id
tch.sal
```

Size of structure and union variables

```c
#include <stdio.h>

struct Stechno
{
  int comp_id;
  float sal;
  double id;
};

union Utechno
{
  int comp_id;
  float sal;
  double id;
};

void main()
{
  struct Stechno stch;
  union Utechno utch;

  printf("The size of structure variable: %d\n", sizeof(stch));
  printf("The size of union variable: %d\n\n", sizeof(utch));

}
```

```
[z2019024@CSLinux Lecture_codes]$ ./LC_struct_16
The size of structure variable: 16
The size of union variable: 8
```

Size of a structure variable is the total size of all the members.
Here, 4 byte (for int) + 4 bytes (for floats) + 8 bytes (for double) = 16 bytes

Size of a union variable is the size of largest member. Here, 8 bytes (for double)

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

Union usage

- Variable format input records (coded records)
- Sharing an area to save storage usage
- Unions not used nearly as much as structures

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Thank you!

The University of
Nottingham

UNITED KINGDOM · CHINA · MALAYSIA