Unit-5 Structure

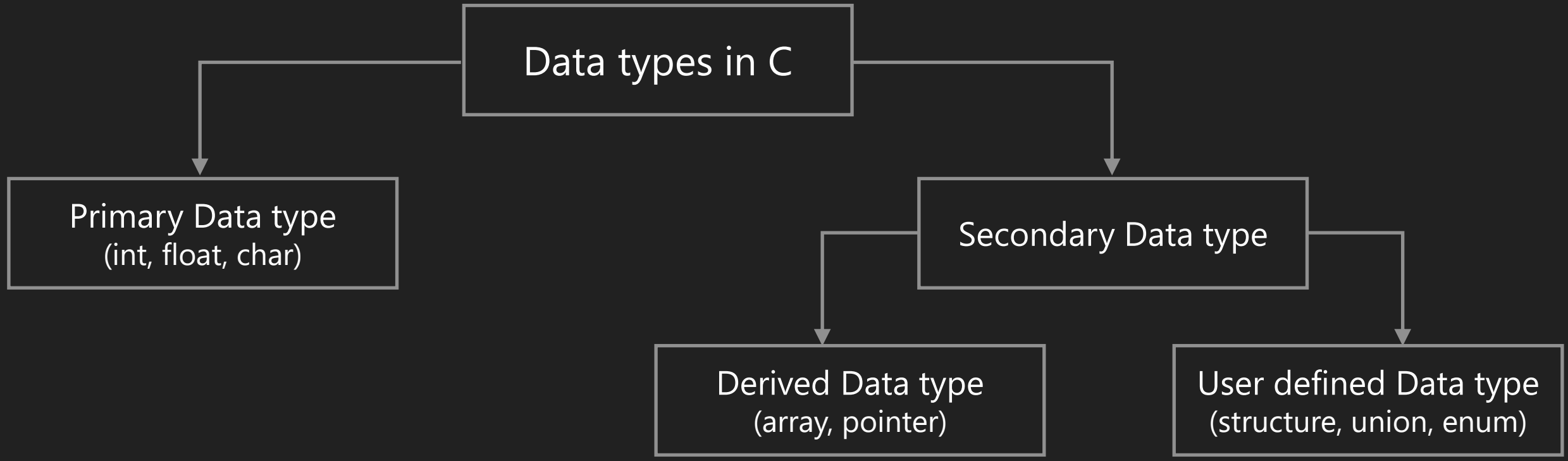
USING

{C}

Programming

Data Types

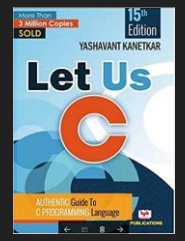
- Data types are defined as the data storage format that a variable can store a data.



- C language has built-in datatypes like primary and derived data types.
- But, still not all real world problems can be solved using those data types.
- We need custom datatype for different situation.

User Defined Datatype

- We need combination of various datatypes to understand different entity/object.
- **Example-1:**
  - **Book**



Title: Let Us C	**Datatype**: char / string
Author: Yashavant Kanetkar	**Datatype**: char / string
Page: 320	**Datatype**: int
Price: 255.00	**Datatype**: float

- **Example-2:**
  - **Student**

Name: ABC	**Datatype**: char / string
Roll_No: 180540107001	**Datatype**: int
CPI: 7.46	**Datatype**: float
Backlog: 01	**Datatype**: int

What is Structure?

- Structure is a collection of logically related data items of different datatypes grouped together under single name.
- Structure is a user defined datatype.
- Structure helps to build a complex datatype which is more meaningful than an array.
- But, an array holds similar datatype record, when structure holds different datatypes records.
- Two fundamental aspects of Structure:
  - Declaration of Structure Variable
  - Accessing of Structure Member

Syntax to Define Structure

- To define a structure, we need to use **`struct`** keyword.
- This keyword is reserved word in C language. We can only use it for structure and its object declaration.

Syntax

```
1 struct structure_name
2 {
3     member1_declaration;
4     member2_declaration;
5     . . .
6     memberN_declaration;
7 };
```

structure_name is name of custom type

memberN_declaration is individual member declaration

- Members can be normal variables, pointers, arrays or other structures.
- Member names within the particular structure must be distinct from one another.

Example to Define Structure

Example

```
1  struct student
2  {
3      char name[30]; // Student Name
4      int roll_no; // Student Roll No
5      float CPI; // Student CPI
6      int backlog; // Student Backlog
7  };
```

- You must terminate structure definition with semicolon ;.
- You cannot assign value to members inside the structure definition, it will cause compilation error.

Example

```
1  struct student
2  {
3      char name[30] = “ABC”; // Student Name
4       . . .
5  };
```

Create Structure variable

- A data type defines various properties about data stored in memory.
- To use any type we must declare its variable.
- Hence, let us learn how to create our custom structure type objects also known as structure variable.
- In C programming, there are two ways to declare a structure variable:
  1. Along with structure definition
  2. After structure definition

Create Structure Variable – Cont.

1. Declaration along with the structure definition

Syntax

```
struct structure_name
{
    member1_declaration;
    member2_declaration;
        . . .
    memberN_declaration;
} structure_variable;
```

Example

```
struct student
{
    char name[30]; // Student Name
    int roll_no; // Student Roll No
    float CPI; // Student CPI
    int backlog; // Student Backlog
} student1;
```

Create Structure Variable – Cont.

2. Declaration after Structure definition

Syntax

```
struct structure_name structure_variable;
```

Example

```
struct student
{
    char name[30]; // Student Name
    int roll_no; // Student Roll No
    float CPI; // Student CPI
    int backlog; // Student Backlog
};
struct student student1; // Declare structure variable
```

Access Structure member (data)

- Structure is a complex data type, we cannot assign any value directly to it using assignment operator.
- We must assign data to individual structure members separately.
- C supports two operators to access structure members, using a structure variable.
  1. Dot/period operator (.)
  2. Arrow operator (->)

Access Structure member (data) – Cont.

1. Dot/period operator (.)
   - It is known as member access operator. We use dot operator to access members of simple structure variable.

Syntax

```
structure_variable.member_name;
```

Example

```
// Assign CPI of student1
student1.CPI = 7.46;
```

2. Arrow operator (->)
   - In C language it is illegal to access a structure member from a pointer to structure variable using dot operator.
   - We use arrow operator to access structure member from pointer to structure.

Syntax

```
pointer_to_structure->member_name;
```

Example

```
// Student1 is a pointer to student type
student1 -> CPI = 7.46;
```

Write a program to read and display student information using structure.

Program

```c
#include <stdio.h>
struct student
{
      char name[40]; // Student name
      int roll; // Student enrollment
      float CPI; // Student mobile number
      int backlog;
};
int main()
{
      struct student student1; // Simple structure variable
      // Input data in structure members using dot operator
      printf("Enter Student Name:");
      scanf("%s", student1.name);
      printf("Enter Student Roll Number:");
      scanf("%d", &student1.roll);
      printf("Enter Student CPI:");
      scanf("%f", &student1.CPI);
      printf("Enter Student Backlog:");
      scanf("%d", &student1.backlog);
      // Display data in structure members using dot operator
      printf("\nStudent using simple structure variable.\n");
      printf("Student name: %s\n", student1.name);
      printf("Student Enrollment: %d\n", student1.roll);
      printf("Student CPI: %f\n", student1.CPI);
      printf("Student Backlog: %i\n", student1.backlog);
}
```

Output

```
Enter Student Name:aaa
Enter Student Roll Number:111
Enter Student CPI:7.89
Enter Student Backlog:0

Student using simple structure variable.
Student name: aaa
Student Enrollment: 111
Student CPI: 7.890000
Student Backlog: 0
```

Write a program to declare time structure and read two different time period and display sum of it.

Program

```c
#include<stdio.h>
struct time {
     int hours;
     int minutes;
     int seconds;
};
int main() {
     struct time t1,t2;
     int h, m, s;
     //1st time
     printf ("Enter 1st time.");
     printf ("\nEnter Hours: ");
     scanf ("%d",&t1.hours);
     printf ("Enter Minutes: ");
     scanf ("%d",&t1.minutes);
     printf ("Enter Seconds: ");
     scanf ("%d",&t1.seconds);
     printf ("The Time is
     %d:%d:%d",t1.hours,t1.minutes,t1.seconds);
     //2nd time
     printf ("\n\nEnter the 2nd time.");
     printf ("\nEnter Hours: ");
     scanf ("%d",&t2.hours);
     printf ("Enter Minutes: ");
     scanf ("%d",&t2.minutes);
     printf ("Enter Seconds: ");
     scanf ("%d",&t2.seconds);
     printf ("The Time is
     %d:%d:%d",t2.hours,t2.minutes,t2.seco
     nds);
     h = t1.hours + t2.hours;
     m = t1.minutes + t2.minutes;
     s = t1.seconds + t2.seconds;
     printf ("\nSum of the two time's is
     %d:%d:%d",h,m,s);
     return 0;
}
```

Output

```
Enter 1st time.
Enter Hours: 1
Enter Minutes: 20
Enter Seconds: 20
The Time is 1:20:20

Enter the 2nd time.
Enter Hours: 2
Enter Minutes: 10
Enter Seconds: 10
The Time is 2:10:10
Sum of the two time's is 3:30:30
```

Array of Structure

- It can be defined as the collection of multiple structure variables where each variable contains information about different entities.
- The array of structures in C are used to store information about multiple entities of different data types.

Syntax

```
struct structure_name
{
    member1_declaration;
    member2_declaration;
    ...
    memberN_declaration;
} structure_variable[size];
```

Write a program to read and display N student information using array of structure.

Program

```c
#include<stdio.h>
struct student {
     char name[20];
     int rollno;
     float cpi;
};
int main( ) {
     int i,n;
     printf("Enter how many records u want to store : ");
     scanf("%d",&n);
     struct student sarr[n];
     for(i=0; i<n; i++)
     {
          printf("\nEnter %d record : \n",i+1);
          printf("Enter Name : ");
          scanf("%s",sarr[i].name);
          printf("Enter RollNo. : ");
          scanf("%d",&sarr[i].rollno);
          printf("Enter CPI : ");
          scanf("%f",&sarr[i].cpi);
     }
     printf("\n\tName\tRollNo\tMarks\t\n");
     for(i=0; i<n; i++) {
          printf("\t%s\t\t%d\t\t%.2f\t\n", sarr[i].name,
          sarr[i].rollno, sarr[i].cpi);
     }
     return 0;
}
```

Output

```
Enter how many records u want to store : 3

Enter 1 record :
Enter Name : aaa
Enter RollNo. : 111
Enter CPI : 7.89

Enter 2 record :
Enter Name : bbb
Enter RollNo. : 222
Enter CPI : 7.85

Enter 3 record :
Enter Name : ccc
Enter RollNo. : 333
Enter CPI : 8.56

    Name    RollNo  Marks
    aaa     111     7.89
    bbb     222     7.85
    ccc     333     8.56
```

Write a program to declare time structure and read two different time period and display sum of it using function.

Program

```c
#include<stdio.h>
struct Time {
     int hours;
     int minutes;
     int seconds;
};
struct Time input(); // function declaration
int main()
{
     struct Time t;
     t=input();
     printf("Hours : Minutes : Seconds\n %d : %d :
     %d",t.hours,t.minutes,t.seconds);
     return 0;
}
struct Time input() // function definition
{
     struct Time tt;
     printf ("Enter Hours: ");
     scanf ("%d",&tt.hours);
     printf ("Enter Minutes: ");
     scanf ("%d",&tt.minutes);
     printf ("Enter Seconds: ");
     scanf ("%d",&tt.seconds);
     return tt; // return structure variable
 }
```

Output

```
Enter Hours: 1
Enter Minutes: 20
Enter Seconds: 20
Hours : Minutes : Seconds
1 : 20 : 20
```

SELF REFERENTIAL STRUCTURE

Self Referential structures are those structures that have one or more pointers which point to the same type of structure, as their member. In other words, structures pointing to the same type of structures are self-referential in nature.

Syntax:

```
 struct structname
{
Datatype member1;
  Datatype member2;
  . . .
Datatype member *n;
};
```

Example:

```
struct node
{
int data1;
char data2;
struct node* link;
};
int main()
{
 struct node ob;
 return 0;
}
```

typedef in C

The **typedef** is a keyword used in C programming to provide some meaningful names to the already existing variable in the C program. It behaves similarly as we define the alias for the commands. In short, we can say that this keyword is used to redefine the name of an already existing variable.

```
#include <stdio.h>
int main()
{
typedef unsigned int unit;
unit i,j;
i=10;
j=20;
printf("Value of i is :%d",i);
printf("\nValue of j is :%d",j);
return 0;
}
```

OUTPUT:-

Value of i is :10 Value of j is :20

Structure using Pointer

- Reference/address of structure object is passed as function argument to the definition of function.

Program

```c
#include <stdio.h>
struct student {
     char name[20];
     int rollno;
     float cpi;
};
int main()
{
     struct student *studPtr, stud1;
     studPtr = &stud1;
     printf("Enter Name: ");
     scanf("%s", studPtr->name);
     printf("Enter RollNo: ");
     scanf("%d", &studPtr->rollno);
     printf("Enter CPI: ");
     scanf("%f", &studPtr->cpi);
     printf("\nStudent Details:\n");
     printf("Name: %s\n", studPtr->name);
     printf("RollNo: %d", studPtr->rollno);
     printf("\nCPI: %f", studPtr->cpi);
     return 0;
}
```

Output

```
Enter Name: ABC
Enter RollNo: 121
Enter CPI: 7.46

Student Details:
Name: ABC
RollNo: 121
CPI: 7.460000
```

Nested Structure

- When a structure contains another structure, it is called nested structure.
- For example, we have two structures named Address and Student. To make Address nested to Student, we have to define Address structure before and outside Student structure and create an object of Address structure inside Student structure.

Syntax

```
struct structure_name1
{
    member1_declaration;
    member2_declaration;
    ...
    memberN_declaration;
};
struct structure_name2
{
    member1_declaration;
    member2_declaration;
    ...
    struct structure1 obj;
};
```

Write a program to read and display student information using nested of structure.

Program

```c
#include<stdio.h>
struct Address
{
    char HouseNo[25];
    char City[25];
    char PinCode[25];
};
struct Student
{
    char name[25];
    int roll;
    float cpi;
    struct Address Add;
};
int main()
{
    int i;
    struct Student s;
    printf("\n\tEnter Student Name : ");
    scanf("%s",s.name);
    printf("\n\tEnter Student Roll Number : ");
    scanf("%d",&s.roll);
    printf("\n\tEnter Student CPI : ");
    scanf("%f",&s.cpi);
    printf("\n\tEnter Student House No : ");
    scanf("%s",s.Add.HouseNo);
    printf("\n\tEnter Student City : ");
    scanf("%s",s.Add.City);
    printf("\n\tEnter Student Pincode : ");
    scanf("%s",s.Add.PinCode);
    printf("\nDetails of Students");
    printf("\n\tStudent Name : %s",s.name);
    printf("\n\tStudent Roll Number :
    %d",s.roll);
    printf("\n\tStudent CPI : %f",s.cpi);
    printf("\n\tStudent House No :
    %s",s.Add.HouseNo);
    printf("\n\tStudent City :
    %s",s.Add.City);
    printf("\n\tStudent Pincode :
    %s",s.Add.PinCode);
    return 0;
    }
```

Output

```
Details of Students
        Student Name : aaa
        Student Roll Number : 111
        Student CPI : 7.890000
        Student House No : 39
        Student City : rajkot
        Student Pincode : 360001
```

Practice Programs

1. Define a structure data type called time_struct containing three member's integer hours, minutes, second. Develop a program that would assign values to individual member and display the time in following format : HH:MM:SS
2. WAP to create structure of book with book title, author name, publication, and price. Read data of n books and display them.
3. Define a structure Person that would contain person name, date of joining, and salary using this structure to read this information of 5 people and print the same on screen.
4. Define a structure time_struct containing three member's integer hour, integer minute and integer second. WAP that would assign values to the individual number and display the time in the following format: 16: 40: 51.
5. Define a structure cricket that will describe the following information:

   Player name

   Team name

   Batting average
6. Using cricket, declare an array player with 50 elements and WAP to read the information about all the 50 players and print team wise list containing names of players with their batting average.
7. Define a structure student_record to contain name, branch, and total marks obtained. WAP to read data for 10 students in a class and print them.

Union

USING

{C}

Programming

What is Union?

- Union is a user defined data type similar like Structure.
- It holds different data types in the same memory location.
- You can define a `union` with various members, but only one member can hold a value at any given time.
- Union provide an efficient way of using the same memory location for multiple-purpose.

Syntax to Define and Access Union

- Declaration of union must start with the keyword union followed by the union name and union's member variables are declared within braces.

Syntax

```
union union_name
{
    member1_declaration;
    member2_declaration;
. . .
    memberN_declaration;
};
```

union_name is name of custom type.

memberN_declaration is individual member declaration.

- Accessing the union members:
  - You need to create an object of union to access its members.
  - Object is a variable of type union. Union members are accessed using the dot operator(.) between union's object and union's member name.

Syntax

```
union union_name union_variable;
```

Example to Define Union

Example

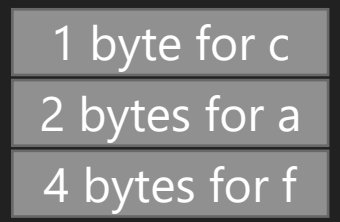
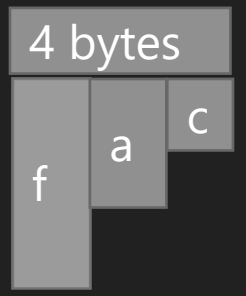
```
union student
{
    char name[30]; // Student Name
    int roll_no; // Student Roll No
    float CPI; // Student CPI
    int backlog; // Student Backlog
} student1;
```

- You must terminate union definition with semicolon ;.
- You cannot assign value to members inside the union definition, it will cause compilation error.

Example

```
union student
{
    char name[30] = “ABC”; // Student Name
. . .
} student1;
```

Structure Vs. Union

COMPARISON	STRUCTURE	UNION
Basic	The separate memory location is allotted to each member of the structure.	All members of the 'union' share the same memory location.
keyword	'struct'	'union'
Size	Size of Structure = sum of size of all the data members.	Size of Union = size of the largest member.
Store Value	Stores distinct values for all the members.	Stores same value for all the members.
At a Time	A structure stores multiple values, of the different members, of the structure.	A union stores a single value at a time for all members.
Declaration	`struct ss { int a; float f; char c };` 	`union uu { int a; float f; char c };` 

Where Union should be used?

- Mouse Programming
- Embedded Programming
- Low Level System Programming

Thank you