Aspirevision Tech™
Education Private Limited

C PROGRAMMING LANGUAGE

BY SHIVAM KUMAR

www.mtaeducation.in

C STRUCTURE

A struct (or structure) is a collection of variables (can be of different types) under a single name.

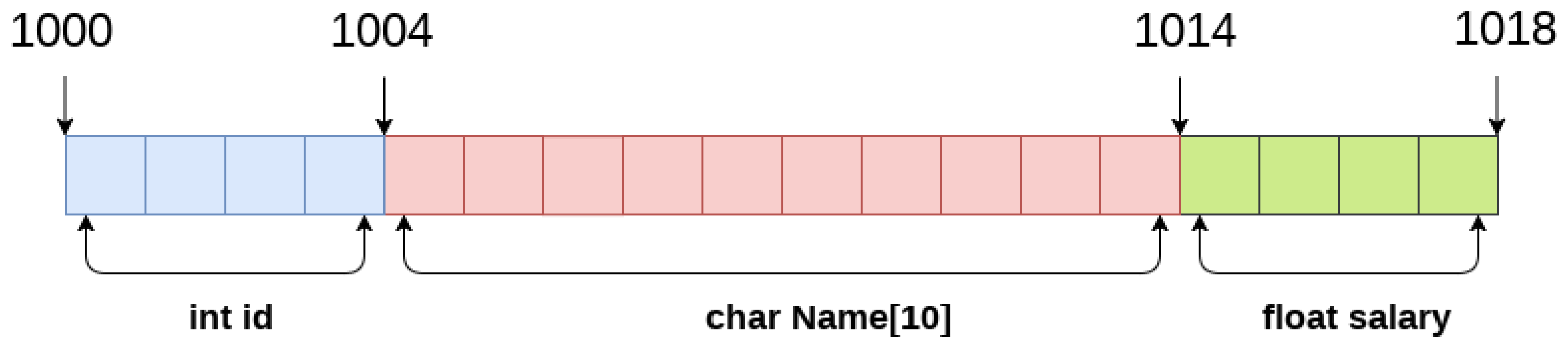
Before you can create structure variables, you need to define its data type. To define a struct, the struct keyword is used.

Syntax of struct:

```
struct structureName
{
  dataType member1;
  dataType member2;
   ...
};
```

C STRUCTURE

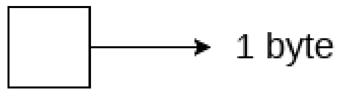
```
struct employee
{   int id;
    char name[20];
    float salary;
};
```



```
struct Employee
{
     int id;
     char Name[10];
     float salary;
} emp;
```

sizeof (emp) = 4 + 10 + 4 = 18 bytes

where;
sizeof (int) = 4 byte
sizeof (char) = 1 byte
sizeof (float) = 4 byte



www.mtaeducation.in

DECLARING STRUCTURE VARIABLE

We can declare a variable for the structure so that we can access the member of the structure easily

- ➢ By struct keyword within main() function
- ➢ By declaring a variable at the time of defining the structure

DECLARING STRUCTURE VARIABLE

1st way:
To declare the structure variable by struct keyword. It should be declared within the main function.

```
struct employee
{   int id;
    char name[50];
    float salary;
};
```

Now write given code inside the main() function.

```
struct employee e1, e2;
```

DECLARING STRUCTURE VARIABLE

2nd way:

Let's see another way to declare variable at the time of defining the structure.

```
struct employee
{   int id;
    char name[50];
    float salary;
}e1,e2;
```

ACCESSING MEMBERS OF THE STRUCTURE

Two ways to access structure members:

- By . (member or dot operator)
- By -> (structure pointer operator

ACCESSING MEMBERS OF THE STRUCTURE

```
#include<stdio.h>
#include <string.h>
struct employee
{   int id;
    char name[50];
}e1;  //declaring e1 variable for structure
int main( )
{
   //store first employee information
   e1.id=101;
   strcpy(e1.name, "Sonoo Jaiswal");//copying string into char array
   //printing first employee information
   printf( "employee 1 id : %d\n", e1.id);
   printf( "employee 1 name : %s\n", e1.name);
return 0;
}
```

KEYWORD TYPEDEF

We use the typedef keyword to create an alias name for data types. It is commonly used with structures to simplify the syntax of declaring variables.

```
struct Distance
{
   int feet;
   float inch;
 };

 int main()
 {
     struct Distance d1, d2;
}
```

```
typedef struct Distance
{
     int feet;
     float inch;
} distances;

 int main()
{
     distances d1, d2;
 }
```

KEYWORD TYPEDEF

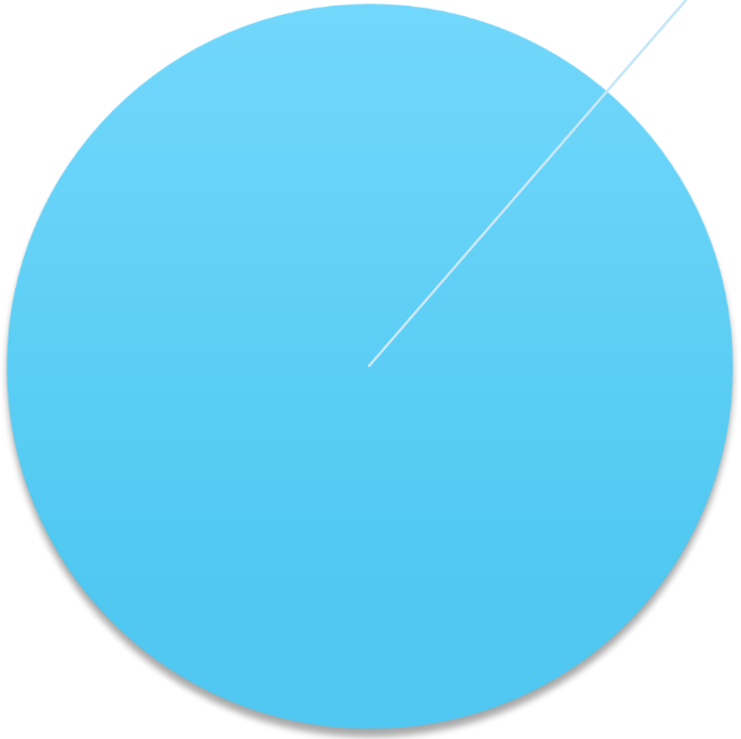
```
#include <stdio.h>
#include <string.h>
// struct with typedef person
typedef struct Person {
  char name[50];
  int citNo;
  float salary;
} person;

int main() {
  // create  Person variable
  person p1;
  strcpy(p1.name, "George Orwell");
  p1.citNo = 1984;
  p1. salary = 2500;
  printf("Name: %s\n", p1.name);
  printf("Citizenship No.: %d\n", p1.citNo);
  printf("Salary: %.2f", p1.salary);
  return 0;
}
```

C STRUCTS AND POINTERS

- To access members of a structure using pointers, we use the -> operator

```
#include <stdio.h>
 struct person
{
   int age;
   float weight;
};
 int main()
 {
    struct person *personPtr, person1;
    personPtr = &person1;
    printf("Enter age: ");
    scanf("%d", &personPtr->age);
    printf("Enter weight: ");
    scanf("%f", &personPtr->weight);
    printf("Displaying:\n");
    printf("Age: %d\n", personPtr->age);
    printf("weight: %f", personPtr->weight);
   return 0; }
```

WHY STRUCTS IN C?



A better approach would be to have a collection of all related information under a single name Person structure and use it for every person.

UNION IN C

A union is a user-defined type similar to structs in C except for one key difference.

Structures allocate enough space to store all their members, whereas **unions can only hold one member value at a time.**

CREATE UNION VARIABLES

When a union is defined, it creates a user-defined type. However, no memory is allocated. To allocate memory for a given union type and work with it, we need to create variables.

```
union car
{
  char name[50];
  int price;
};

int main()
{
  union car car1, car2, *car3;
  return 0;
}
```

```
union car
{
   char name[50];
   int price;
} car1, car2, *car3;
```

CREATE UNION VARIABLES

```
#include <stdio.h>
union unionJob
 {
    //defining a union
    char name[32];
    float salary;
     int workerNo;
 } uJob;
struct structJob
 {
    char name[32];
    float salary;
    int workerNo;
} sJob;
int main()
 {
      printf("size of union = %d bytes", sizeof(uJob));
      printf("\nsize of structure = %d bytes", sizeof(sJob));
return 0;
}
```