Programming with C and C++

CSC-101 (Lecture 25)

Dr. R. Balasubramanian
Professor
Department of Computer Science and Engineering
Mehta Family School of Data Science and Artificial Intelligence
Indian Institute of Technology Roorkee
Roorkee 247 667

bala@cs.iitr.ac.in
https://faculty.iitr.ac.in/cs/bala/

Function Pointers

```
1   #include<stdio.h>
2   int *larger(int *, int *);
3
4   int main() {
5       int a = 10, b = 15;
6       int *greater;
7       // passing address of variables to function
8       greater = larger(&a, &b);
9       printf("Larger value = %d", *greater);
10      return 0;
11  }
12
```

https://ideone.com/6TUUGe

Function Pointers

```
13  int *larger(int *a, int *b) {
14      if (*a > *b) {
15          return a;
16      }
17      // returning address of greater value
18      return b;
19  }
20
```

stdout

```
Larger value = 15
```

Function Pointers

```c
#include <stdio.h>

int *arraymanip()
{
  static int arr[7];
  printf("Enter the elements in an array : ");
  for(int i=0;i<7;i++)
  {
      scanf("%d",&arr[i]);
  }
  for(int i=0;i<7;i++)
  {
      arr[i]*=2;
  }
  return arr;
}

```

https://ideone.com/5EHf3J

```c
int main()
{
   int *ptr;
   ptr=arraymanip();
   printf("\nThe manipulated elements are :");
   for(int i=0;i<7;i++)
   {
      printf("%d ", ptr[i]);
   }
}
```

stdin

```
1 2 3 4 5 6 7
```

stdout

```
Enter the elements in an array :
The manipulated elements are :2 4 6 8 10 12 14
```

Structures in C

- Structure is another user defined data type available in C that allows to combine data items of different kinds.

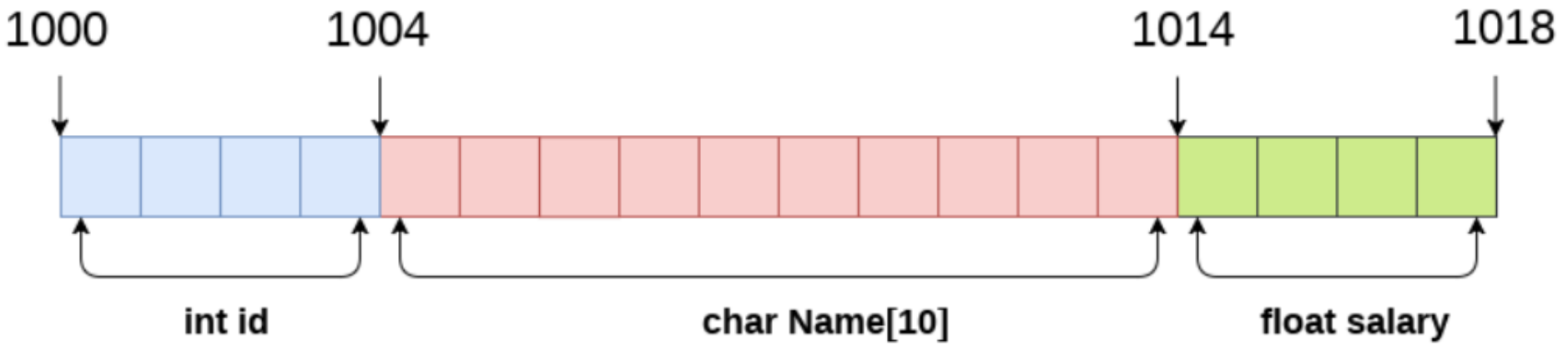
```
struct [structure tag] {

   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```

Example 1

```
struct Books {
    char  title[50];
    char  author[50];
    char  subject[100];
    int   book_id;
} book;
```

Example 2

```
struct employee
{    int id;
     char name[10];
     float salary;
};
```



Declaring structure variables

- There are two ways to declare structure variable:
  - By **`struct`** keyword within **`main()`** function
  - By declaring a variable at the time of defining the structure.

1.
```
struct employee
{    int id;
     char name[50];
     float salary;
};
```

Now write given code inside the **`main()`** function.

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

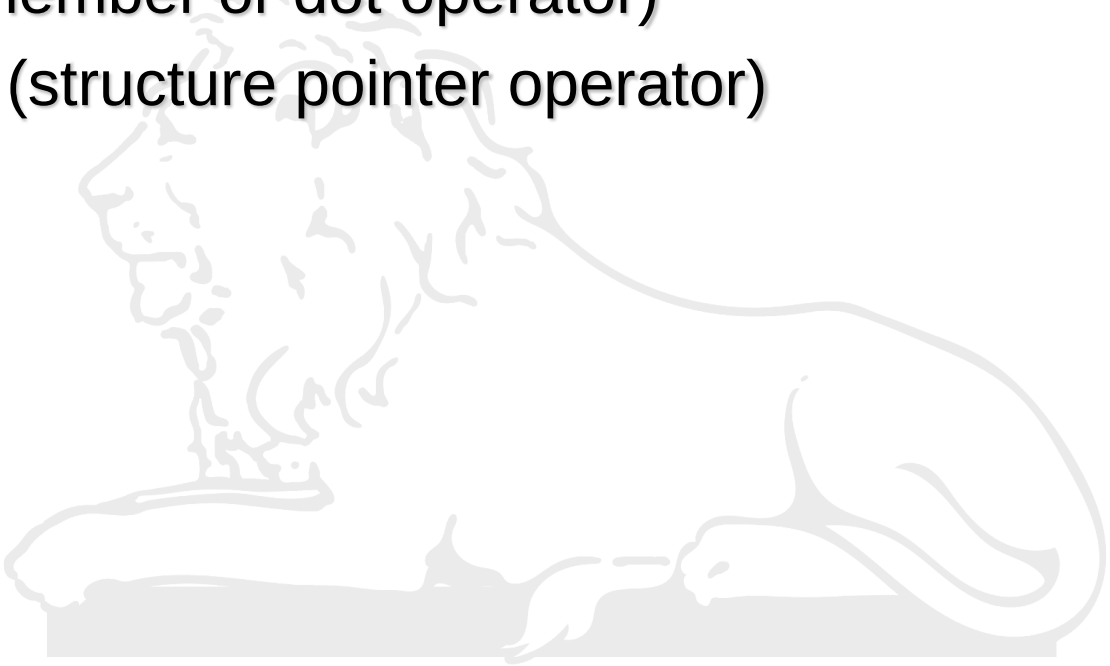
Declaring structure variables

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

- If number of variables are not fixed, use the 1st approach. It provides you the flexibility to declare the structure variable many times.
- If number of variables are fixed, use 2nd approach. It saves your code to declare a variable in **`main()`** function.

Accessing members of the structure

- There are two ways to access structure members:
  - By . (member or dot operator)
  - By --> (structure pointer operator)



Structures in C

```
1   #include<stdio.h>
2   #include <string.h>
3
4   struct employee
5   {    int id;
6        char name[50];
7   } e1;  //declaring e1 variable for structure
8
```

https://ideone.com/y7p3H5

```
 9  int main( )
10  {
11     //store first employee information
12     e1.id=1;
13     strcpy(e1.name, "KL Rahul");//copying string into char array
14     //printing first employee information
15     printf( "employee 1 id : %d\n", e1.id);
16     printf( "employee 1 name : %s\n", e1.name);
17  return 0;
18  }
19
```

```
Success #stdin #stdout 0s 5412KB
employee 1 id : 1
employee 1 name : KL Rahul
```

Size of struct variable

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

struct employee
{    int id;
     char name[50];
     float salary;
} e1;  //declaring e1 variable for structure

int main( )
{
   printf( "Size of the structure employee: %d\n", sizeof(e1));

return 0;
}
```

https://ideone.com/LuJxAi

```
Success #stdin #stdout 0.01s 5460KB
Size of the structure employee: 60
```

- Why does the structure's area differ from each member's?

  - The size of a structure is not always equal to the sum of the sizes of its members.
  - Compilers may add padding between members to ensure that they are aligned properly in memory.
  - Alignment requirements can vary depending on the CPU architecture and compiler options.
  - The amount of padding between members can affect the structure's size.

Thank You!