

Structures



a collection of values of **different** data types

EXAMPLE

For a student store the following :

name (String)

roll no (Integer)

cgpa (Float)

Syntax

```
struct student {  
    char name[100];  
    int roll;  
    float cgpa;  
};
```

```
struct student s1;  
s1.cgpa = 7.5;
```

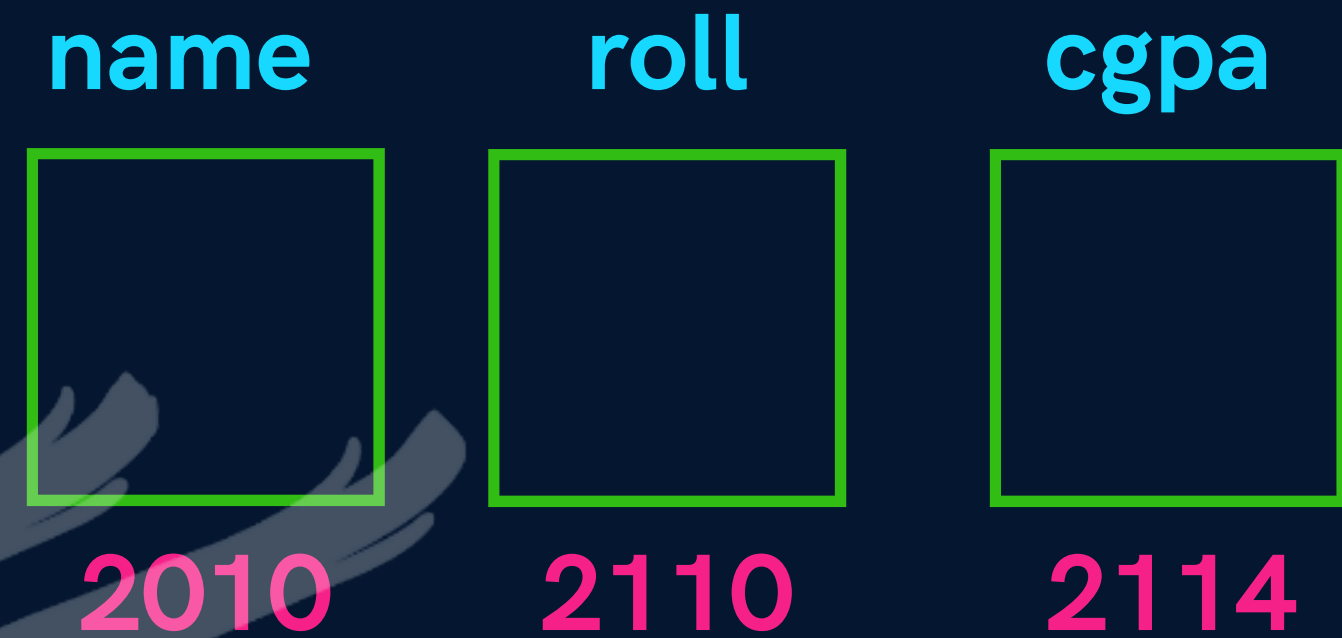
Syntax

```
struct student {  
    char name[100];  
    int roll;  
    float cgpa;  
}
```



Structures in Memory

```
struct student {  
    char name[100];  
    int roll;  
    float cgpa;  
}
```



structures are stored in contiguous memory locations

Benefits of using Structures

- Saves us from creating too many variables
- Good data management/organization

Array of Structures

```
struct student ECE[100];
```

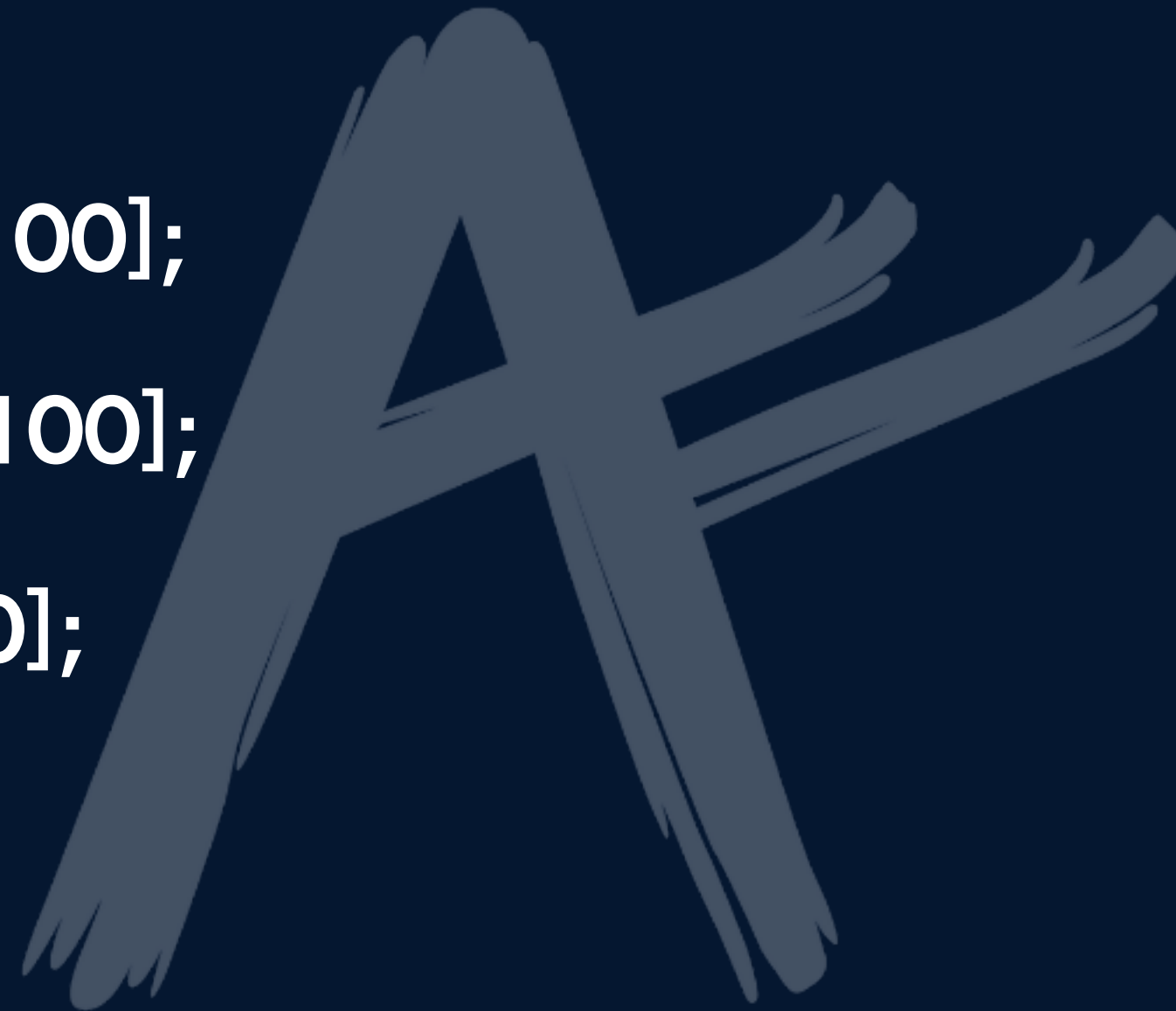
```
struct student COE[100];
```

```
struct student IT[100];
```

ACCESS

```
IT[0].roll = 200;
```

```
IT[0].cgpa = 7.6;
```



Initializing Structures

```
struct student s1 = { "shradha", 1664, 7.9};
```

```
struct student s2 = { "rajat", 1552, 8.3};
```

```
struct student s3 = { 0 };
```

Pointers to Structures

```
struct student s1;
```

```
struct student *ptr;
```

```
ptr = &s1;
```



Arrow Operator

`(*ptr).code` \longleftrightarrow `ptr->code`



Passing structure to function

//Function Prototype

void printInfo(struct student s1);

typedef Keyword



used to create **alias** for data types

```
typedef struct ComputerEngineeringStudent{  
    int roll;  
    float cgpa;  
    char name[100];  
} coe;
```

coe student1;