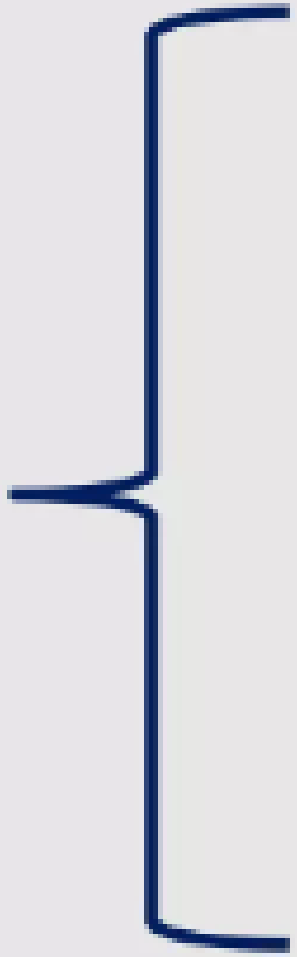
Structures in C++

What is a Structure?

- A structure is a collection of variables under a single name. These variables can be of different data types, and each has a name that is used to select it from the structure
- There is always a requirement in most of our data processing applications that the relevant should be grouped and handled as a group
- In structure , we introduce a new data type
- A structure can contain any data type including array and another structure as well

- It provides a simple method of abstraction and grouping
- Each variable declared inside structure is called member of structure
- A structure may itself contain structures
- A structure can be assigned to, as well as passed to, and returned from functions
- We declare a structure using the keyword ***struct***

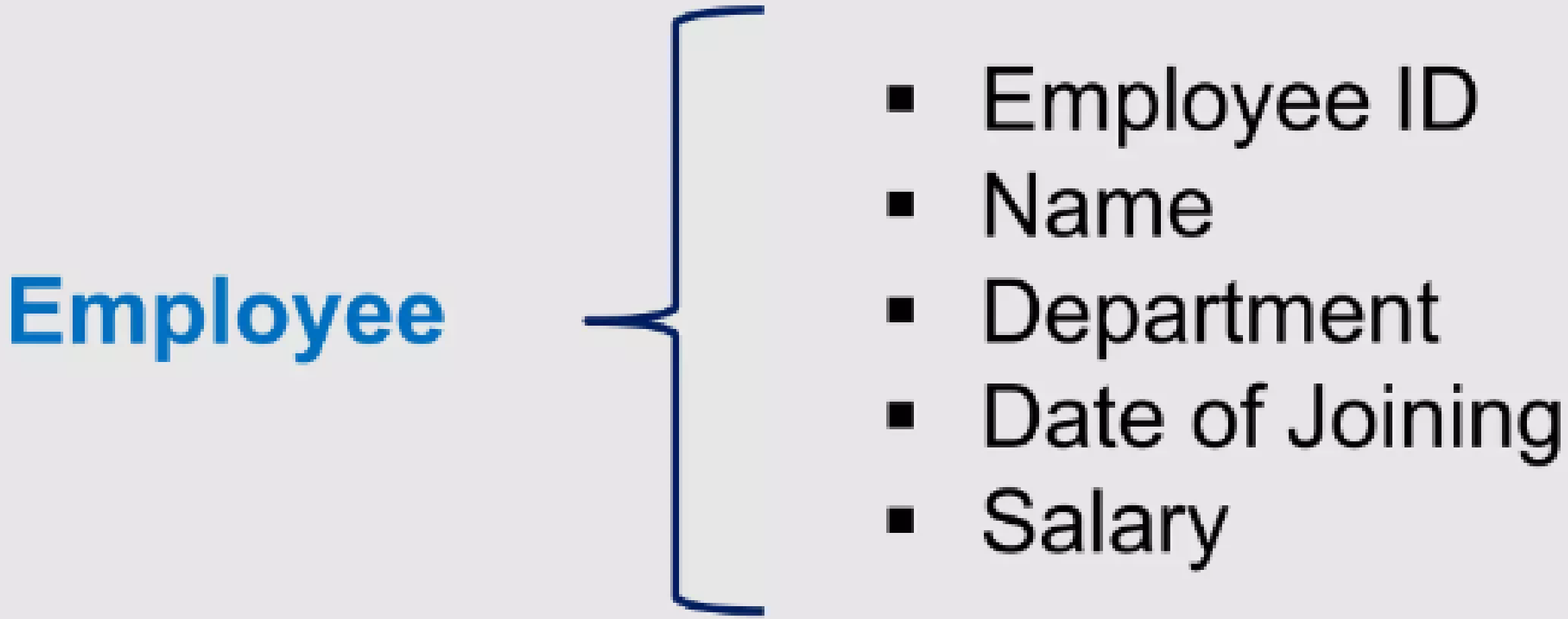
Students



- Name
- Address
- Date of Birth
- CGPA
- Discipline

Car

- Model
- Manufacturer company
- Engine size
- Number of seats


Employee
Employee ID
Name
Department
Date of Joining
Salary

Steps to Create Structure

- Declare Structure
- Initialize Members of Structure
- Access Structure Elements

Declaration of a Structure

- ***Struct*** keyword is used for creating a structure

Structure Declaration Ways

- ✓ By struct keyword
- ✓ By declaring variables at the time of defining structure

Declaration of a Structure

- The structure is declared by using the keyword ***struct*** followed by ***structure name***, also called a ***tag***.
- Then the ***structure members*** (variables) are defined with their type and variable names *inside the open and close braces* { and }
- Finally, the closed braces *end with a semicolon* denoted as**;** following the statement.
- The above structure declaration is also called a *Structure Specifier*

- Structures are syntactically declare with:
  - ✓ Keyword *struct*
  - ✓ Followed by the *name of the structure*
  - ✓ The data, contained in the structure, is defined in the curly braces
- All the variables that we have been using can be part of structure

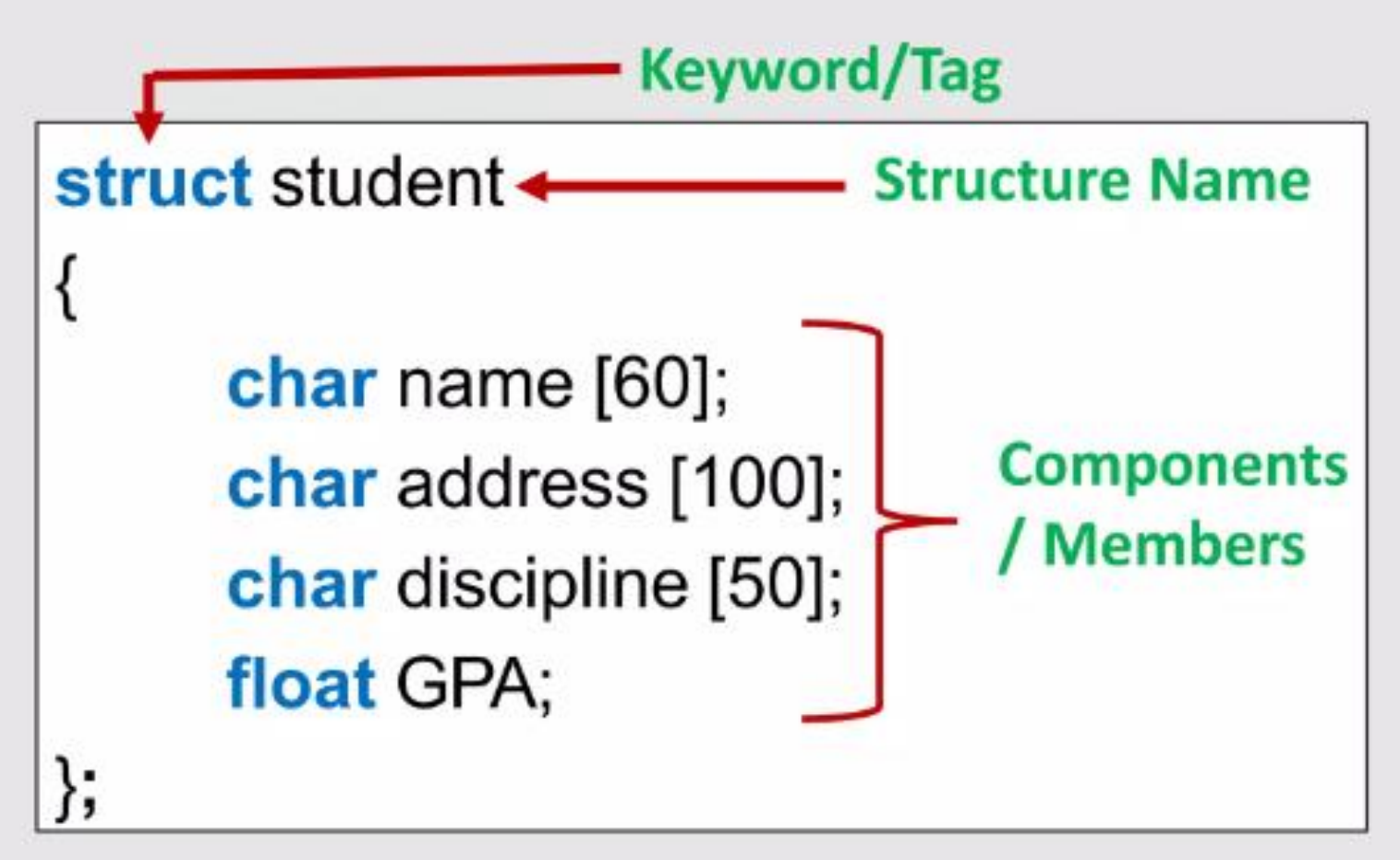
Declaration of a Structure

```
struct type_name
    {
        member_type1 member_name1;
        member_type2 member_name2;
        member_type3 member_name3;
        .
        .
        .



    }
```

Approach 1 (Declaration of a Structure)

```
Struct student
    {
    char name[40];
    char address[60];
    char discipline[50];
    float GPA;
};
```

Approach 1 (Declaration of a Structure)



- Student is called the structure tag, and is your brand new data type, like int, double or char
- Name, address, discipline and GPA are structure members
- Note: Memory is not allocated at the time of its declaration. Memory is allocated when we declare structure variable

Approach 1 (Declaration of a Structure)

```
struct student
    {
    string name;
    string address;
    string discipline;
    float GPA;
};
```

Approach 1 (Declaration of a Structure)

```
struct car
    {
    string model;
    string company;
    string engineSize;
    int  price;
};
```

Approach 1 (Declaration of a Structure)

```
struct employee
    {
    string employeeID;
    string name;
    string department;
    float  salary;
};
```

Declaring Variables of Type struct

- The most efficient method of dealing with structure variables is to define the structure *globally*.
- To declare a structure globally, place it BEFORE int main().
- The structure variables can the be defined locally in main, for example.

```
. . .
struct STUDENT_TYPE
{
      string name, street, city, state, zipcode;
      int age;
      double ID_num;
      double grade;
};

int main()
{
      // declare two variables of the new type
      STUDENT_TYPE student1, student2;


            . . .
 }
```

Approach 2 (Declaring Variables of Type struct)

```
struct STUDENT_TYPE
{
    string name, street, city, state, zipcode;
    int age;
    double ID_num;
    double grade;
}

student1, student2;
```

Accessing Structure Members

- To access any members of a structure, we use the **member access operator (.)**.
- The member access operator is coded as period between the structure variable name and the structure member that we wish to access
- Remember we would use ***struct*** keyword to define variables of structure type.

Accessing Structure Members

- Suppose, you want to access age of structure variable **student1** and assign it 50 to it.
- We can perform this task by using following code below:

```
student1.age = 50
```

- Taking input as:

```
cin>>student1.age;
```

Example 1

C++ Program to assign data to members of a structure variable and display it

```
#include<iostream>
using namespace std;
struct Person
{
    char name[40];
    int age;
    float salary;
};
int main ()
{
    Person p1;
    cout<<"Enter full name";
    cin.get(p1.name, 40);
    cout<<"Enter age : ";
    cin>>p1.age;
    cout<<"Enter salary";
    cin>>p1.salary;
//Displaying user Entered information
cout<<"\n Displaying Information"<<endl;
cout<<"Name : "<<p1.name<<endl;
cout<<"Age : "<<p1.age<<endl;
cout<<"Salary : "<<p1.salary<<endl;
return 0;
}
```

Output

Enter Full name : Rajab Hamisi

Enter age : 22

Enter salary : 200000

Displaying user Entered Information

Name : Rajab Hamisi

Age : 22

Salary : 200000

```
#include <iostream>
#include <cstring>

 using namespace std;

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

int main() {
   struct Books Book1;       // Declare Book1 of type Book
   struct Books Book2;       // Declare Book2 of type Book

    // book 1 specification
   strcpy( Book1.title, "Learn C++ Programming");
   strcpy( Book1.author, "Chand Miyan");
   strcpy( Book1.subject, "C++ Programming");
   Book1.book_id = 6495407;

   // book 2 specification
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Yakit Singha");
   strcpy( Book2.subject, "Telecom");
   Book2.book_id = 6495700;

  // Print Book1 info
   cout << "Book 1 title : " << Book1.title <<endl;
   cout << "Book 1 author : " << Book1.author <<endl;
   cout << "Book 1 subject : " << Book1.subject <<endl;
   cout << "Book 1 id : " << Book1.book_id <<endl;

  // Print Book2 info
   cout << "Book 2 title : " << Book2.title <<endl;
   cout << "Book 2 author : " << Book2.author <<endl;
   cout << "Book 2 subject : " << Book2.subject <<endl;
   cout << "Book 2 id : " << Book2.book_id <<endl;

   return 0;
}
```

- When the above code is compiled and executed, it produces the following result:

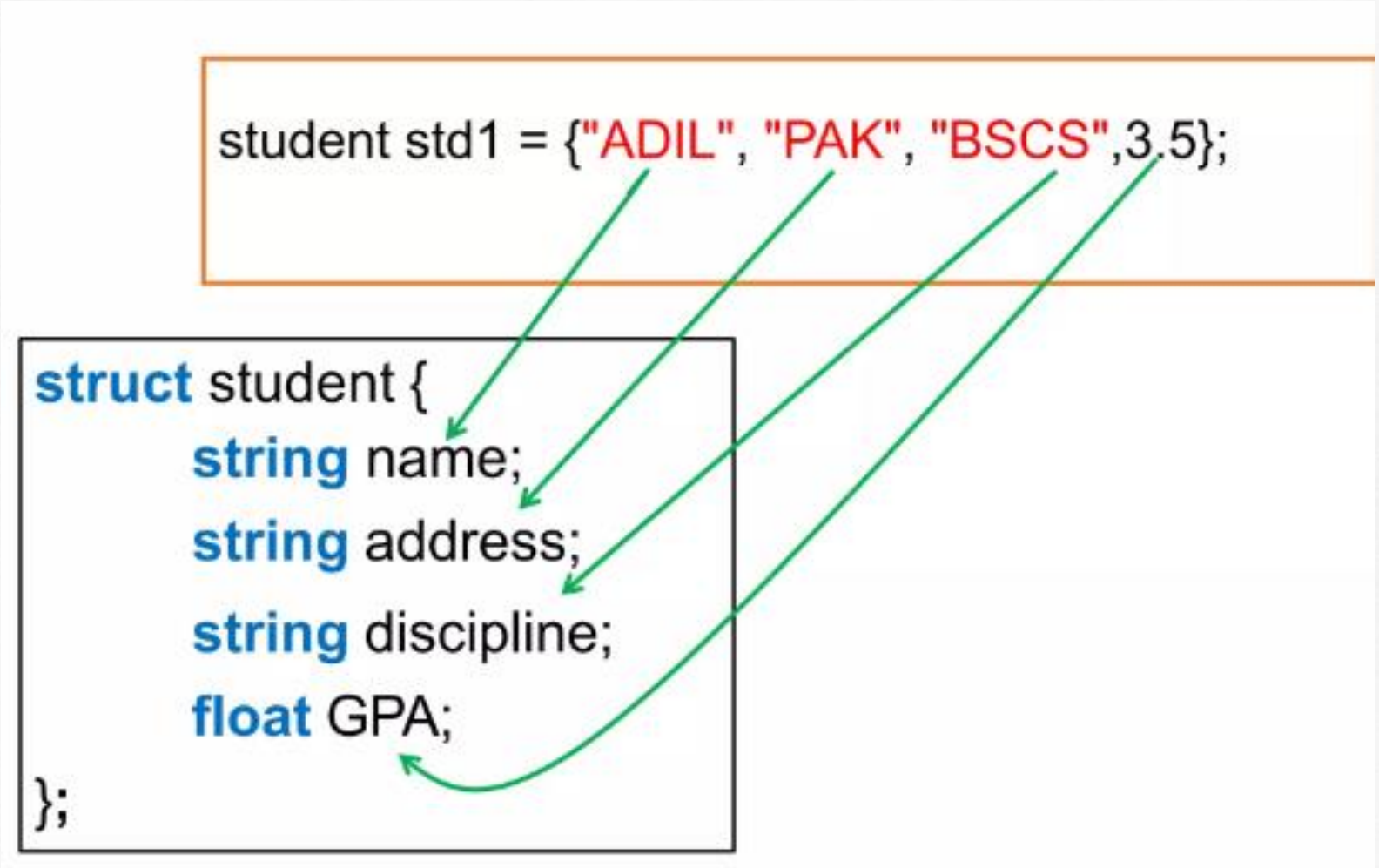
```
Book 1 title : Learn C++ Programming

Book 1 author : Chand Miyan

Book 1 subject : C++ Programming

Book 1 id : 6495407

Book 2 title : Telecom Billing

Book 2 author : Yakit Singha

Book 2 subject : Telecom

Book 2 id : 6495700
```

Initializing Structures

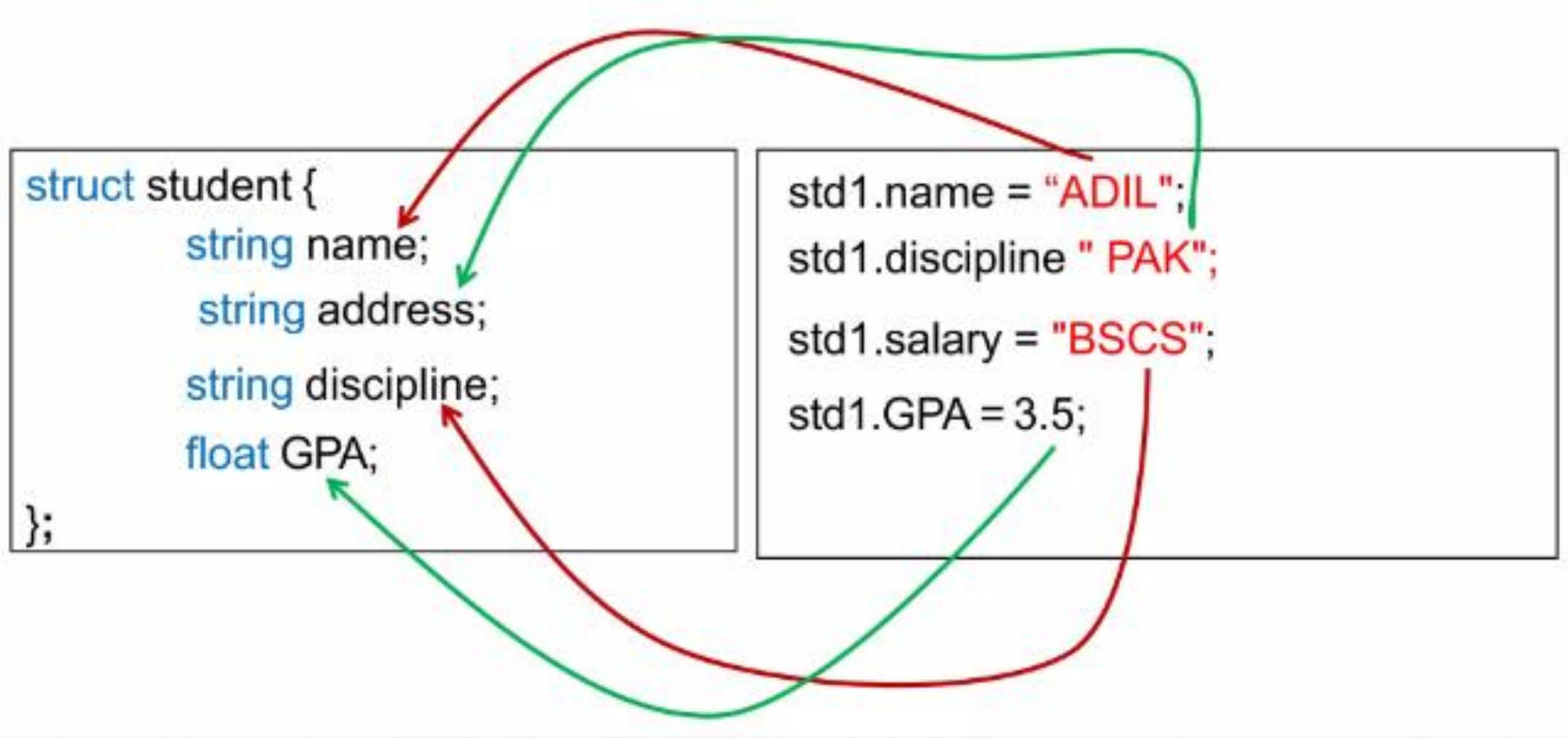
- Like normal variables structures can be initialized at the time of declaration. Initialization of structure is almost similar to initializing array. The structure object is followed by equal sign and the list of values enclosed in braces and each value is separated with comma.
- Example

  Person p1 = {" Rajab Hamisi“, 22, 200000}

Initializing Structures (first way)



Initializing Structures (second way)



```
//Example
#include<iostream>
using namespace std;
struct Student {
    string Name;
    string Address;
    string Programme;
    float GPA;
};
int main()
{
Student S = {" Rajab Hamisi ","MBY “,BEEE“, 4.5};
cout<<“\n Student Name :“<<S.Name;
cout<<“\n Student Address :“<< S.Address;
cout<<“\n Student Programme :“<<S.Programme;
cout<<“\n Student GPA :“<<S.GPA;
}
```

```
// Declare a structure named "car"
struct car {
  string brand;
  string model;
  int year;
};
int main() {
  // Create a car structure and store it in myCar1;
  car myCar1;
  myCar1.brand = "BMW";
  myCar1.model = "X5";
  myCar1.year = 1999;
  // Create another car structure and store it in myCar2;
  car myCar2;
  myCar2.brand = "Ford";
  myCar2.model = "Mustang";
  myCar2.year = 1969;
   // Print the structure members
  cout << myCar1.brand << " " << myCar1.model << "
" << myCar1.year << "\n";
  cout << myCar2.brand << " " << myCar2.model << "
" << myCar2.year << "\n";
   return 0;
}
```

Structure Variable in Assignment Statement

```
S1 = S2;
```

The statement assigns the value of each member of S2 to the corresponding member of S1. Note that one structure variable can be assigned to another only when they are of the same structure type, otherwise compiler will give an error

Problem Statement

Write a program which declares two variables of a structure and copies the contents of first variable into the second variable.

```
#include<iostream>
using namespace std;
struct employee
{
    int id;
    string name;
    double salary;
    string address;
};
int main ( )
{
employee emp1 = {1,
"Sikoro",200000,"Mbeya"};
employee emp2 = emp1;
cout<<"Employee 1 Information"<<endl;
cout<<"-----------------------------"<<endl;
cout<<"ID : \t\t\t"<<emp1.id<<endl;
cout<<"Name : \t\t"<<emp1.name<<endl;
cout<<"Salary : \t\t\"<<emp1.salaryendl;
cout<<"Address : \t"<<emp1.address<<endl;
cout<<"_____________________________"<<endl;
cout<<"Employee 2 Information"<<endl;
cout<<"-----------------------------"<<endl;
cout<<"ID : \t\t\t"<<emp2.id<<endl;
cout<<"Name : \t\t"<<emp2.name<<endl;
cout<<"Salary : \t\t\"<<emp2.salaryendl;
cout<<"Address : \t"<<emp2.address<<endl;
cout<<"_____________________________"<<endl;
return 0;
}
```

Exercise

Write a C++ Program to Store Information (name, registration number and marks) of a Student Structure and store information

```
#include<iostream>
using namespace std;
struct student
{
    char name[60]
    int regno;
    float marks;
};
int main ( )
{
struct student s;
cout<<"Enter Information of Student "<<endl;
cout<<"Enter name : ";
cin>>s.name;
cout<<"Enter Registration Number : ";
cin>>s.regno;
cout<<"Enter Marks : ";
cin>>s.marks;
cout<<"\n Displaying Information "<<endl;
cout<<"\n Name : " <<s.name;
cout<<"\n Reg. Number : " <<s.regno;
cout<<"\n Marks : " <<s.marks;
return 0;
}
```

Structures as Function Arguments

- You can pass a structure as a function argument in very similar way as you pass any other variable or pointer.
- You would access structure variables in the similar way as you have accessed in the above example

```
#include <iostream>
#include <cstring>

 using namespace std;
void printBook( struct Books book );

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

int main() {
   struct Books Book1;    // Declare Book1 of type Book
   struct Books Book2;    // Declare Book2 of type Book

    // book 1 specification
   strcpy( Book1.title, "Learn C++ Programming");
   strcpy( Book1.author, "Chand Miyan");
   strcpy( Book1.subject, "C++ Programming");
   Book1.book_id = 6495407;

   // book 2 specification
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Yakit Singha");
   strcpy( Book2.subject, "Telecom");
   Book2.book_id = 6495700;

    // Print Book1 info
   printBook( Book1 );

   // Print Book2 info
   printBook( Book2 );

   return 0;
}
void printBook( struct Books book ) {
   cout << "Book title : " << book.title <<endl;
   cout << "Book author : " << book.author <<endl;
   cout << "Book subject : " << book.subject <<endl;
   cout << "Book id : " << book.book_id <<endl;
}
```

- When the above code is compiled and executed, it produces the following result:

```
Book title : Learn C++ Programming

Book author : Chand Miyan

Book subject : C++ Programming

Book id : 6495407

Book title : Telecom Billing

Book author : Yakit Singha

Book subject : Telecom

Book id : 6495700
```

Pointers to Structures

- You can define pointers to structures in very similar way as you define pointer to any other variable as follows:

```
struct Books *struct_pointer;
```

- Now, you can store the address of a structure variable in the above defined pointer variable. To find the address of a structure variable, place the & operator before the structure's name as follows:

```
struct_pointer = &Book1;
```

- To access the members of a structure using a pointer to that structure, you must use the -> operator as follows:

```
struct_pointer->title;
```

- Let us re-write above example using structure pointer, hope this will be easy for you to understand the concept:

```
#include <iostream>
#include <cstring>

 using namespace std;
void printBook( struct Books *book );

struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};
int main() {
   struct Books Book1;   // Declare Book1 of type Book
   struct Books Book2;  // Declare Book2 of type Book

    // Book 1 specification
   strcpy( Book1.title, "Learn C++ Programming");
   strcpy( Book1.author, "Chand Miyan");
   strcpy( Book1.subject, "C++ Programming");
   Book1.book_id = 6495407;

   // Book 2 specification
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Yakit Singha");
   strcpy( Book2.subject, "Telecom");
   Book2.book_id = 6495700;

    // Print Book1 info, passing address of structure
   printBook( &Book1 );

   // Print Book1 info, passing address of structure
   printBook( &Book2 );

   return 0;
}

// This function accept pointer to structure as parameter.
void printBook( struct Books *book ) {
   cout << "Book title : " << book->title <<endl;
   cout << "Book author : " << book->author <<endl;
   cout << "Book subject : " << book->subject <<endl;
   cout << "Book id : " << book->book_id <<endl;
}
```

- When the above code is compiled and executed, it produces the following result:

```
Book title : Learn C++ Programming
Book author : Chand Miyan
Book subject : C++ Programming
Book id : 6495407
Book title : Telecom Billing
Book author : Yakit Singha
Book subject : Telecom
Book id : 6495700
```