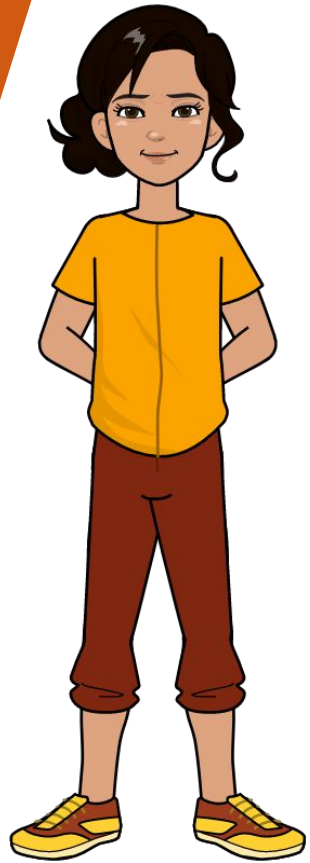
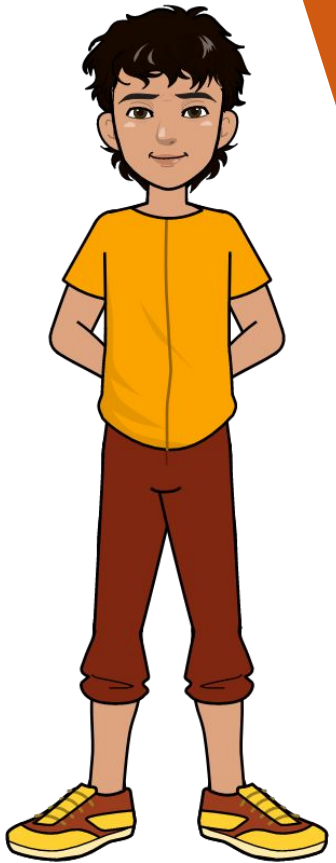


Structures



Review: Data types

- Scalar variables can store only one value at a time.

```
#include<iostream>
using namespace std;

main(){
    int number = 15;
    float float_data = 6.5;
    string word = "Hello";
    char character = 'A';
}
```

Review: Data types

- Arrays can store more than one value, but of same data type.

```
#include<iostream>
using namespace std;

main(){
    int num[5] = {5,4,11,1,6};
    int cars[5][5] = {{10, 7, 12, 10, 4},
                      {18, 11, 15, 17, 2},
                      {23, 19, 12, 16, 14},
                      {7, 12, 16, 0, 2},
                      {3, 5, 6, 2, 1}};
}
```

Review: Data types

- What if we want to store the information of a student?
- Information contains Student Name (string type), Roll Number(int type) and GPA (float type).

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

Review: Data types

- We can store such information in Parallel arrays.

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

Review: Data types

- Is this an Efficient Solution?

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

Review: Data types

- This is not an efficient solution, because the information is scattered in 3 different arrays, only linked through indexes.

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

User-defined Data types

- C++ provides us an opportunity to define our own data type according to the requirement.

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

Structure

- This Data-type is called **Structure**. We can create our own data type to group items of possibly different types into a single type.

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

Structure

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

- Syntax to define a Data Type is:

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

Structure

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

- Syntax to define a Data Type is:

Keyword

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

Structure

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

- Syntax to define a **Data Type** is:

Keyword

Name of user-defined data type

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

Structure

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

- Syntax to define a Data Type is:

Keyword

Name of user-defined data type

Within these curly brackets we can declare more than one variable

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

Structure

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

- Syntax to define a **Data Type** is:

Keyword

Name of user-defined data type

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

Within these curly brackets we can declare more than one variable

Semi-colon to show structure ends

Structure

Name	Roll Number	GPA
Jack	2312	3.9
John	1111	3.2
Ibrahim	2121	3.4

- Syntax to define a **Data Type** is:

Keyword

Name of user-defined data type

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

Within these curly brackets we can declare more than one variable

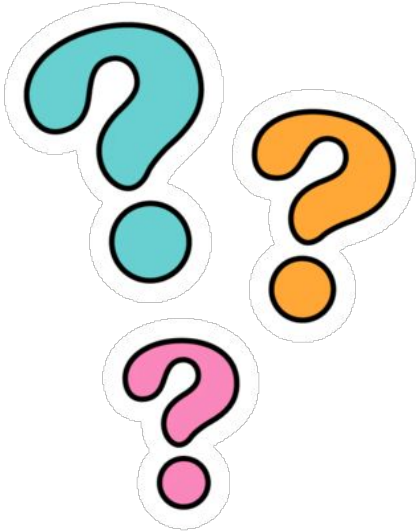
These variables are called members

Semi-colon to show structure ends

Structure: How to declare

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- We have defined our own data type.
- Now the question is how to declare the variable of data type **student**.



Structure: How to declare

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- We will declare the student variable just as we declare other int, float and char variables

```
student stu1;
```

Structure: How to declare

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- We will declare the student variable just as we declare other int, float and char variables

```
student stu1;
```

datatype

Structure: How to declare

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- We will declare the student variable just as we declare other int, float and char variables

```
student stu1;
```

datatype

Variable name for student type

Structure: How to access

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- The members of structure variable are accessed using a dot (.) operator.

```
student stu1;
```

Structure: How to access

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- The members of structure variable are accessed using a dot (.) operator.

```
student stu1;
stu1.name = "Jack";
```

Structure: How to access

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- The members of structure variable are accessed using a dot (.) operator.

```
student stu1;
stu1.name = "Jack";
stu1.rollNumber = 2312;
```

Structure: How to access

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- The members of structure variable are accessed using a dot (.) operator.

```
student stu1;
stu1.name = "Jack";
stu1.rollNumber = 2312;
stu1.cgpa = 3.9;
```

Structure: How to access

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- We can declare and initialize as many variables as we like.

```
student stu1;
stu1.name = "Jack";
stu1.rollNumber = 2312;
stu1.cgpa = 3.9;
```

```
student stu2;
stu2.name = "John";
stu2.rollNumber = 1111;
stu2.cgpa = 3.2;
```

```
student stu3;
stu3.name = "Ibrahim";
stu3.rollNumber = 2121;
stu3.cgpa = 3.4;
```

Structure: Input from User

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- We can even take input from the user in the structure variable, pass to functions and return from the functions

Structure: Input from User

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- Let's make a function that takes input from the user in the student structure

```
void input()
{
    student stu;
    cout << "Enter Student Name:" << endl;
    cin.ignore();
    getline(cin, stu.name);
    cout << "Enter Student Roll Number:" << endl;
    cin >> stu.rollNumber;
    cout << "Enter Student CGPA:" << endl;
    cin >> stu.cgpa;
}
```

Structure: return

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- Let's **return** the student structure from the function.

```
void input()
{
    student stu;
    cout << "Enter Student Name:" << endl;
    cin.ignore();
    getline(cin, stu.name);
    cout << "Enter Student Roll Number:" << endl;
    cin >> stu.rollNumber;
    cout << "Enter Student CGPA:" << endl;
    cin >> stu.cgpa;
}
```

Structure: return

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- Let's **return** the student structure from the function.

```
student input()
{
    student stu;
    cout << "Enter Student Name:" << endl;
    cin.ignore();
    getline(cin, stu.name);
    cout << "Enter Student Roll Number:" << endl;
    cin >> stu.rollNumber;
    cout << "Enter Student CGPA:" << endl;
    cin >> stu.cgpa;
    return stu;
}
```

Structure: return

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- Let's make an array of 3 students.

```
student input()
{
    student stu;
    cout << "Enter Student Name:" << endl;
    cin.ignore();
    getline(cin, stu.name);
    cout << "Enter Student Roll Number:" << endl;
    cin >> stu.rollNumber;
    cout << "Enter Student CGPA:" << endl;
    cin >> stu.cgpa;
    return stu;
}
```

Structure: return

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- Let's make an array of 3 students.

```
student input()
{
    student stu;
    cout << "Enter Student Name:" << endl;
    cin.ignore();
    getline(cin, stu.name);
    cout << "Enter Student Roll Number:" << endl;
    cin >> stu.rollNumber;
    cout << "Enter Student CGPA:" << endl;
    cin >> stu.cgpa;
    return stu;
}
```

```
main()
{
    student total_stu[3];
    for (int i = 0; i < 3; i++)
    {
        total_stu[i] = input();
    }
}
```

Structure: parameter passing

```
struct student
    {
        string name;
        int rollNumber;
        float cgpa;
    };
```

- Let's print all the records of 3 students.

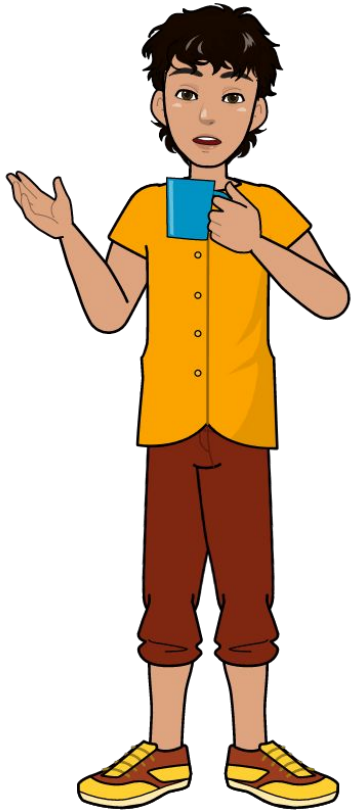
```
student input()
{
    student stu;
    cout << "Enter Student Name:" << endl;
    cin.ignore();
    getline(cin, stu.name);
    cout << "Enter Student Roll Number:" << endl;
    cin >> stu.rollNumber;
    cout << "Enter Student CGPA:" << endl;
    cin >> stu.cgpa;
    return stu;
}
```

```
void print(student stu)
{
    cout << stu.name;
    cout << "\t";
    cout << stu.rollNumber;
    cout << "\t";
    cout << stu.cgpa;
    cout << endl;
}
```

```
main()
{
    student total_stu[3];
    for (int i = 0; i < 3; i++)
    {
        total_stu[i] = input();
    }
    for (int i = 0; i < 3; i++)
    {
        print(total_stu[i]);
    }
}
```

Learning Objective

Write a **C++** program to define and use user defined data type (**Struct**).



Conclusion

- Structure is a user defined Data Type.
- It is a collection of variables of different data types under a single name.
- Syntax to define a structure is:

```
struct name
{
    // Different Predefined Datatypes
};
```

- Structure variable is declared the same way as the variables of pre-defined datatypes.
- The members of structure variable are accessed using a dot (.) operator
- Structure variables can be passed to a function and returned in a similar way as normal variables.

Self Assessment:

1. Suppose that you have the following definitions:

```
struct timeType
{
    int hr;
    double min;
    int sec;
};
```

```
struct tourType
{
    string cityName;
    int distance;
    timeType travelTime;
};
```

Self Assessment:

```
struct timeType
{
    int hr;
    double min;
    int sec;
};
```

```
struct tourType
{
    string cityName;
    int distance;
    timeType travelTime;
};
```

A. Declare the variable **destination** of type **tourType**.
B. Write C++ statements to store the following data in **destination**: **cityName:** Chicago, **distance:** 550 miles, **travelTime:** 9 hours and 30 minutes.
C. Write the definition of a function to output that data stored in a variable of type **tourType**.
D. Write the definition of a **value-returning function** that inputs data into a variable of type **tourType**.
E. Write the definition of a **void function** with a reference parameter of type tourType to input data in a variable of type **tourType**.

Self Assessment: (Video Profile Activity)

2. Write a program that declares a struct to store the data of a football player (player's name, player's position, number of touchdowns, number of catches, number of passing yards, number of receiving yards, and the number of rushing yards).

Declare an array of 10 components to store the data of 10 football players. Your program must contain a function to input data and a function to output data. Add functions to search the array to find the index of a specific player, and update the data of a player. (You may assume that the input data is stored in a file.)

Self Assessment:

2. Before the program terminates, give the user the option to save data in a file. Your program should be menu driven, giving the user various choices.