PROBLEM SOLVING AND PROGRAMMING

User defined data types - structs

structures

- Student is an object which contain attributes like rollno, name, department, course, etc.
  - All these attributes of different types.
- How to store such an information under same name ?
  - structures
- A Structure is a collection of related data items, possibly of different types.
- A structure type in C++ is called struct.
- A struct is heterogeneous in that it can be composed of data of different types.
- In contrast, array is homogeneous since it can contain only data of the same type.

structures

- Examples:
  - Student record: student id, name, major, gender, start year, …
  - Bank account: account number, name, currency, balance, …
  - Address book: name, address, telephone number, …
- In database applications, structures are called records.

structures

- Individual components of a struct type are called members (or fields).
- Members can be of different types (simple, array or struct).

```
struct student
{   int rollno;
    char name[20];
    char department[30];
    char course[20];
}s1, s2, s3;
```

Members of the **`struct`**

where student is a structure name // like int (int is a primitive data type)

s1, s2 and s3 are variables of structure student.

Defining a Structure

```
struct structure_name
{
data-type member-1;
data-type member-2;
data-type member-3;
data-type member-4;
};
```

Declaration of Structure Variable

```
struct student
{
    int roll_no;
    string name;
    int phone_number;
};
int main()
{
    struct student p1, p2, p3;
    return 0;
}
```

declare structure variables at the time of defining the structure as follows.

```
struct student
{
    int roll_no;
    std::string name;
    int phone_number;
}p1, p2, p3;
```

Syntax for array within structure

```
struct struct-name
{
 datatype var1; // normal variable
datatype array [size]; // array variable
-  - - - - - - - - - - - - - - - - - - - - - -
datatype varN;
};
struct-name obj;
```

Syntax for declaring structure array

```
struct struct-name
 {
 datatype var1;
 datatype var2;
 - - - - - - - - - - - - - - - - - - - - - -
datatype varN;
};

 struct-name obj [ size ];
```

Declaring **struct** variables

struct student p, q, r;

- Declares and sets aside storage for three variables – p, q, and r – each of type struct student.

struct student M[25];

- Declares a 25-element array of struct student; allocates 25 units of storage, each one big enough to hold the data of one student

struct motor *m;

- Declares a pointer to an object of type struct student

Accessing Members of a **struct**

struct student p;
struct student q[10];
struct motor *r;

- Then

p.rollno	— is the roll no
p.name	— is the name
p.Department	— is the department name
p.course	— is the course name

q[i]. rollno	— is the of the rollno of ith student
q[i]. name	— is the name of the ith student

r -> rollno	— is the rollno of the student pointed to by **r.**
p -> name	— is the name of the student pointed by r.

Operations on **struct**

- Copy/assign

```
struct student p, q;
p = q;
```

- Get address

```
struct student p;
struct student *s;
s = &p;
```

- Access members

```
p.rollno;
s -> rollno;
```

Example for array within structure

```
struct Student
 {
 int Roll;
 char Name[25];
 int Marks[3]; //Statement 1 : array of marks
int Total;
float Avg;
};
void main()
{
int i;
Student S;
 cout << "\n\nEnter Student Roll : ";
cin >> S.Roll;
 cout << "\n\nEnter Student Name : ";
cin >> S.Name;
S.Total = 0;
for(i=0;i<3;i++)
 {
cout << "\n\nEnter Marks " << i+1 << " : ";
 cin >> S.Marks[i];
 S.Total = S.Total + S.Marks[i];
 }
 S.Avg = S.Total / 3;
 cout << "\nRoll : " << S.Roll;
cout << "\nName : " << S.Name;
cout << "\nTotal : " << S.Total;
cout << "\nAverage : " << S.Avg;
}
```

Output :

```
Enter Student Roll : 10
Enter Student Name : Kumar
 Enter Marks 1 : 78
Enter Marks 2 : 89
 Enter Marks 3 : 56


 Roll : 10
Name : Kumar
Total : 223
Average : 74.00000
```

Example

Consider the problem discussed in the previous class and do the solution using structures.

```
#include<iostream>
using namespace std;
struct student
{   int rollno;
    int marks;
};
student s1[60], s2[60], s3[60];
struct total
{   int rollno;
    int m1,m2, m3,sum;
    char grade;
 }t[60];
```

Example

```
int main()
{   int n;
    cout << "Enter number of students:";  cin >> n;
    // read the data into s1, s2, s3
    int i=0;
    // get the data into student structure
    while(i< n)
    {     cout << "\n Enter the " << i+1 << "student details";
           cin >> s1[i].rollno >> s1[i].marks >> s2[i].marks >>  s3[i].marks;
           s2[i].rollno = s3[i].rollno = s1[i].rollno;
           i++;
    }
```

Example

```
// read the data from the student and write it into total
i=0;
while(i<=n) // Include the marks validation
{
  t[i].rollno = s1[i].rollno;
  t[i].m1 = s1[i].marks;
  t[i].m2 = s2[i].marks;
  t[i].m3 = s3[i].marks;
  t[i].sum = t[i].m1 + t[i].m2 + t[i].m3;
  if (t[i].m1 >= 40 && t[i].m2 >= 40 && t[i].m3 >= 40)
    t[i].grade = 'P';
  else  t[i].grade = 'F';
 i++;
}
```

Example

```
    // display the data
    i=0;
    while(i<n)
    {
     cout << endl << t[i].rollno << " " <<  t[i].m1 << " " << t[i].m2 ;
     cout << " " << t[i].m3 << " " ;t[i].sum << t[i].grade;
     i++;
    }
  return 0;
}
```

```
#include <iostream>
using namespace std;
struct student
 {
		char name[50];
		int roll;
		float marks;
 } s[10];
int main()
 {
		cout << "Enter information of students: " << endl; // storing information
		for(int i = 0; i < 10; ++i)
		{
			s[i].roll = i+1;
			cout << "For roll number" << s[i].roll << "," << endl;
			cout << "Enter name: ";
			cin >> s[i].name;
			cout << "Enter marks: ";
			cin >> s[i].marks;
			cout << endl;
		}
cout << "Displaying Information: " << endl; // Displaying
information
for(int i = 0; i < 10; ++i)
 {
cout << "\nRoll number: " << i+1 << endl;
cout << "Name: " << s[i].name << endl;
cout << "Marks: " << s[i].marks << endl;
}
return 0;
}
```

```
Enter information of students:
For roll number1,
Enter name: Tom Enter
marks: 98
For roll number2,
Enter name: Jerry
Enter marks: 89
 . . .


Displaying Information:
Roll number: 1
Name: Tom
Marks: 98
. . .
```

```
int main()
{
  struct student
   {
      int roll_no;
     string name;
     int phone_number;
   };

struct student p1 = {1,"Brown",123443};
struct student p2, p3;
p2.roll_no = 2;
p2.name = "Sam";
p2.phone_number = 1234567822;
p3.roll_no = 3;
p3.name = "Addy";
p3.phone_number = 1234567844;
cout << "First Student" << endl;
cout << "roll no : " << p1.roll_no << endl;
cout << "name : " << p1.name << endl; cout
<< "phone no : " << p1.phone_number <<
endl;
 cout << "Second Student" << endl;
cout << "roll no : " << p2.roll_no << endl;
cout << "name : " << p2.name << endl;
cout << "phone no : " << p2.phone_number
<< endl;
cout << "Third Student" << endl;
cout << "roll no : " << p3.roll_no << endl;
cout << "name : " << p3.name << endl;
cout << "phone no : " << p3.phone_number
<< endl;
 return 0;
}
```

copy two structures

```
    struct student
      {
          int roll_no;
          string name;
          int phone_number;
      };
     int main()
      {
          struct student p1 = {1,"Brown",123443};
          struct student p2;
          p2 = p1;
          cout << "roll no : " << p2.roll_no << endl;
          cout << "name : " << p2.name << endl;
          cout << "phone number : " << p2.phone_number << endl;
          return 0;
    }
```

all the elements of p1 will get copied to p2.

Pointers to Structures

pointer **ptr** to point to the structure variable **stud**. Thus, 'ptr' now stores the address of the structure variable 'stud

```
struct student
 {
    string name;
     int roll_no;
};
int main()
{
    struct student stud = {"Sam",1};
    struct student *ptr;
    ptr = &stud;
    cout << stud.name << stud.roll_no << endl;
    cout << ptr->name << ptr->roll_no << endl;
    return 0;
}
```

Structure to Function

Passing by Value

```
struct student
 {
   int roll_no;
   string name;
   int phone_number;
};
void display(struct student st)
{
  cout << "Roll no : " << st.roll_no << endl;
  cout << "Name : " << st.name << endl;
  cout << "Phone no : " << st.phone_number << endl;
}
```

```
int main()
{
struct student s;
s.roll_no = 4;
s.name = "Ron";
s.phone_number = 888888;
display(s);
return 0;
}
```

Passing by Reference

```
#include <iostream>
#include <cstring>
 using namespace std;
struct student
{
int roll_no;
string name;
int phone_number;
};
void display(struct student *st)
{
cout << "Roll no : " << st -> roll_no << endl;
cout << "Name : " << st -> name << endl;
cout << "Phone no : " << st -> phone_number << endl;
}
```

```
int main()
{
struct student s;
s.roll_no = 4;
s.name = "Ron";
s.phone_number = 888888;
display(&s);
return 0;
}
```