10. Structures and unions.

Defining a structure

```
struct   tag-name
{
  data-type  member1;
  data-type  member2;
   ---        ---
   ---        ---
};
```

Declaring structure variables

1. The keyword struct
2. The structure tagname
3. List of variable names separated by commas
4. A terminating semicolon

Structure initialization

```
main()
{
 struct
 {
  int weight;
  float height;
 } student = {60, 180.75};
 ...
 ...
}
```

Rules for initializing structures

1. we cannot initialize individual members inside the structure template.
2. The order of values enclosed in braces must match the order of members in the structure definition
3. It is permitted to have a partial initialization.

Operations on individual members

```
student1.number ++;
++ student1.number;
```

The precedence of the member operator is higher than all arithmetic and relational operators and therefore no parentheses are required.

The member operator or dot operator is used to manipulated using expressions and operators.

Eg:- float sum = student1.marks + student2.marks
student2.marks * = 0.5;

The unions are concept borrowed from structures and therefore follow the Same syntax as structures. The major difference is in the storage.

In structure → Each member has its own memory

In union → all members of it use same memory.

1) program using nested structures. (student + address)

```
#include<stdio.h>
struct address {
    char street [50];
    char city [50];
    int pin;
};

struct student {
    char name [50];
    int roll;
    struct address addr;
};

int main(){
    struct student s;
    printf ("Enter name:");
    scanf ("%s", s.name);
    printf ("%d", &s.roll);
    printf ("Enter street);
    scanf ("%s", s.addr.street);
  printf("\n student details, \n name=%s Roll=%d street=%s",
         s.name, s.roll, s.addr.street);
    return 0;
}
```

2) Program using array of structures (multiple students)

```
#include<stdio.h>
struct student {
   char name [50];
   float marks;
};
int main(){
   int n,i;
   struct student s[100];
   printf ("enter no.of students:");
   scanf ("%d", &n);
```

```
    for(i=0; i<n; i++){
        printf("Enter name and marks for student %d:", i+1);
        scanf("%s %f", s[i].name, &s[i].marks);
    }
    printf("\nStudent details:\n");
    for(i=0; i<n; i++){
        printf("Name:%s marks:%.2f\n", s[i].name, s[i].marks);
    }
  return 0;
}
```

5. program to append student records using structure

```
#include<stdio.h>
struct student {
    int roll;
    char name[50];
    float marks;
};
int main(){
    struct student s;
    FILE *fp;
    int n,i;
    printf(" Enter no. of students to append:");
    scanf("%d", &n);
    fp = fopen("students.txt", "a");
    for(i=0; i<n; i++){
        printf(" Enter roll, name, marks for student %d", i+1);
        scanf("%d %s %f", &s.roll, s.name, &s.marks);
        fprintf(fp,"%d %s %.2f\n", s.roll, s.name, s.marks);
    }
  fclose(fp);
  printf("updated records\n");
  fp = fopen("students.txt", "r");
  return 0;
}
```

4. program to store employee details using union for salary

```
#include <stdio.h>
union salary{
    int i_salary;
    float f_salary;
};

struct Employee {
    int id;
    char name[50];
    union salary sal;
};

int main() {
    struct Employee e;
    printf("Enter Employee Id :");
    scanf("%d", &e.id);
    printf("Enter name :");
    scanf("%s", e.name);
    printf("Enter salary (as float): ");
    scanf("%f", &e.sal.f_salary);
    printf("Employee details \n Id: %d name: %s salary: %.2f\n",
      e.id, e.name, e.sal.f_salary);
    return 0;
}
```

5. program to store marks in a union.

```
#include <stdio.h>
union marks {
    int imarks;
    float fmarks;
    double dmarks;
};

int main() {
    union marks m;
    m.imarks =85
    printf("Integer marks :%d\n", m.imarks);
    m.fmarks =88.5;
    printf("float marks :%d\n", m.fmarks);
    m.dmarks =85.75;
    printf("double marks :%.2lf\n", m.dmarks);
    return 0;
```

6. program To demonstrate union inside a structure

```
#include<stdio.h>
union salary {
      int i;
      float f;
};
struct employee {
      int id;
      char name [50];
      union salary sal;
};
int main (){
      struct employee e;
      printf("enter id: ");
      scanf ("%d", &e.id);
      printf("Enter name: ");
      scanf ("%s", e.name);
      printf ("Enter salary (float): ");
      scanf ("%f", &e.sal.f);
      printf ("\n Employee details:\n Id: %d name: %s salary: %.2f\n",
       e.id, e.name, e.sal.f);
      return 0;
 }
```

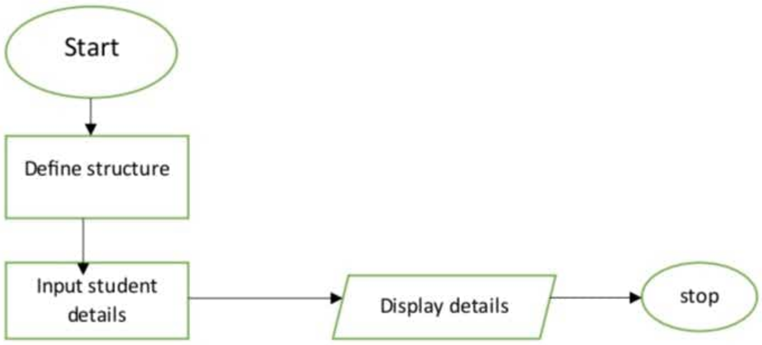
Structures and unions

1. Program to store and display student details using structures

Algorithm

1. Start
2. Define a structure Student with members: name, roll, marks.
3. Declare a variable of type Student.
4. Input values for name, roll, and marks.
5. Display the values.
6. Stop

Flow-chart



2. Program to store and display details of N students

Algorithm

1. Start
2. Define structure Student.
3. Declare an array of Student.
4. Input n (number of students).
5. Loop through and input details for each student.
6. Loop again to display details.
7. Stop

3. Program to find student with highest marks

Algorithm

1. Start
2. Define structure Student.
3. Input n and student details.
4. Initialize maxMarks = s[0].marks.
5. Traverse array: if any student’s marks > maxMarks, update maxMarks and store index.
6. Print the student with highest marks.
7. Stop

4. Program to store and display employee details

Algorithm

1. Start
2. Define structure Employee with id, name, salary.
3. Input employee details.
4. Display details.
5. Stop

5. Program to find employee with highest salary

Algorithm

1. Start
2. Define structure Employee.
3. Input n employees.
4. Initialize maxSalary = e[0].salary.
5. Loop through employees: if salary > maxSalary, update.
6. Print employee with highest salary.
7. Stop

6. Program to sort students by marks

Algorithm

1. Start
2. Define Student structure.
3. Input n and student details.
4. Use a sorting algorithm (bubble sort) to sort by marks.
5. Print sorted list.
6. Stop

7. Program to merge employee records

Algorithm

1. Start
2. Define Employee structure.
3. Input n1 employees into array1.
4. Input n2 employees into array2.
5. Merge both arrays into array3.
6. Display merged list.
7. Stop

8. Program to search a student by roll number

Algorithm

1. Start
2. Define Student structure.
3. Input n students.
4. Input roll number to search.
5. Loop through array, compare roll.
6. If found, display student; else display not found.
7. Stop

9. Program to store and display book details

Algorithm

1. Start
2. Define Book structure with id, title, author, price.
3. Input details of books.
4. Display them.
5. Stop

10. Program to find costliest book

Algorithm

1. Start
2. Define Book structure.
3. Input n books.
4. Initialize max = first book price.
5. Traverse array; if higher price found, update.
6. Print book with max price.
7. Stop

11. Program to search a student by roll number using structure

Algorithm

1. Start
2. Define Student structure with roll, name, marks.
3. Input n students.
4. Input roll number to search.
5. Loop through array and compare roll number.
6. If found, display student details. Else, print "Not found".
7. Stop

12. Program to sort students by marks using structure

Algorithm

1. Start
2. Define Student structure.
3. Input n students.
4. Use bubble sort to sort by marks descending.
5. Display sorted list.
6. Stop

13. Program using nested union (employee bonus and allowance)

Algorithm

1. Start
2. Define union Bonus with amount or percentage.
3. Define union Allowance with travel or medical.
4. Define structure Employee with id, name, bonus (union), allowance (union).
5. Input employee details.
6. Display all details.
7. Stop

14. Program to calculate total salary including bonus using union

Algorithm

1. Start
2. Define union Bonus with amount or percentage.
3. Define structure Employee with id, name, basic salary, bonus (union), bonus type.

4. Input employee details.
5. Calculate total salary (add bonus).
6. Display total salary.
7. Stop

15. Program to find maximum marks using array of structures

Algorithm

1. Start
2. Define Student structure with name and marks.
3. Input n students.
4. Initialize max = first student marks.
5. Loop through array and update max if higher found.
6. Display student with maximum marks.
7. Stop

7. Stop

16. Program to calculate average marks using array of structures

Algorithm

1. Start
2. Define Student structure with name and marks.
3. Input n students.
4. Loop through array and sum marks.
5. Calculate average = total/n.
6. Display average marks.
7. Stop

17. Program to store multiple bonuses using nested union

Algorithm

1. Start
2. Define union Bonus with amount or percentage.
3. Define structure Employee with id, name, basic salary, and array of bonuses (union).
4. Input employee details and multiple bonuses.
5. Calculate total salary including all bonuses.
6. Display total salary.
7. Stop

---

18. Program to calculate total marks for multiple subjects using nested structures

Algorithm

1. Start
2. Define structure Subjects with marks1, marks2, marks3.
3. Define structure Student with name, roll, and subjects.
4. Input n students with marks.
5. Calculate total marks for each student.
6. Display total marks.
7. Stop

19. Program to find employee with maximum salary using union

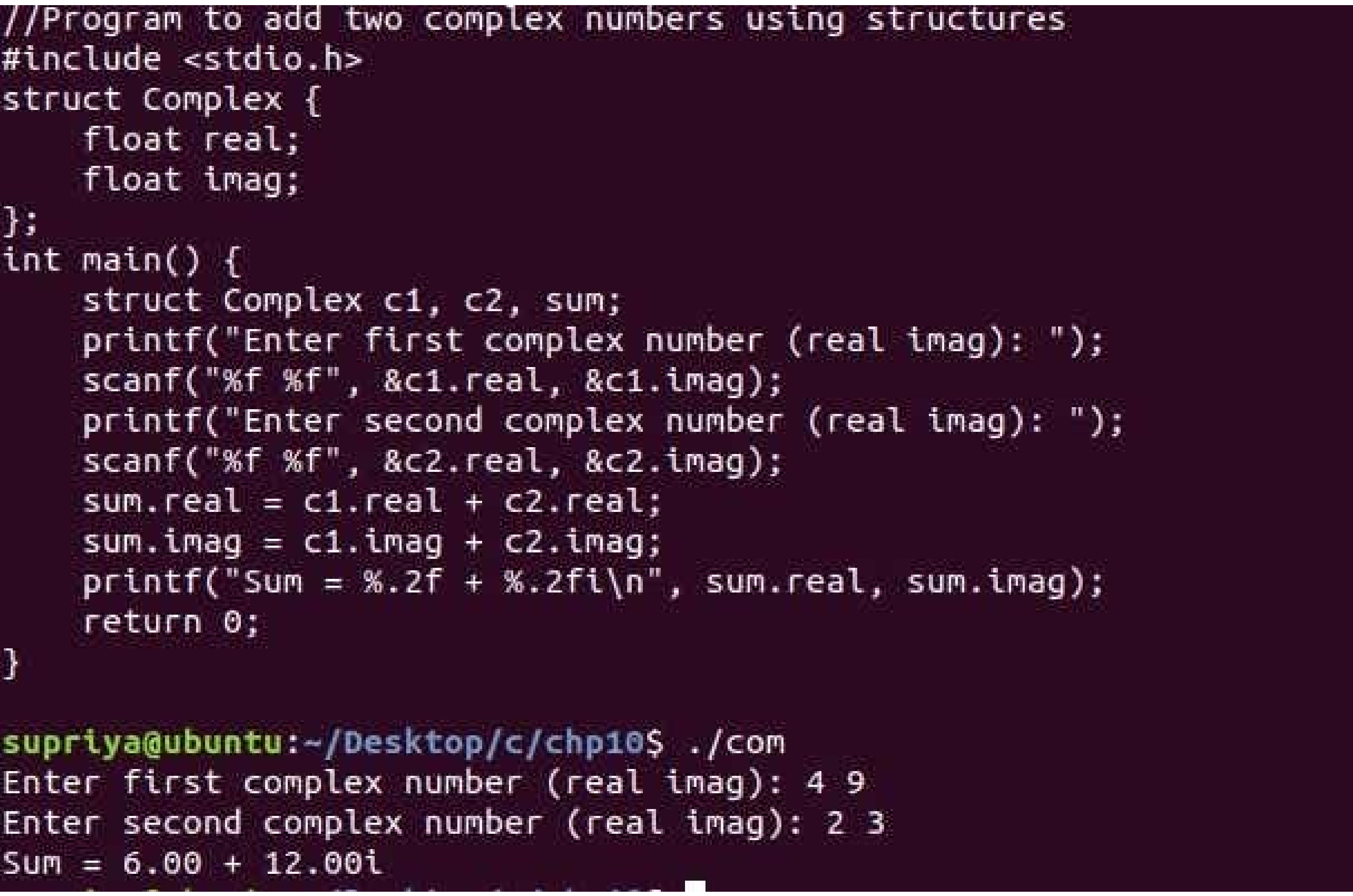
Algorithm

1. Start
2. Define union Salary with monthly or weekly.
3. Define structure Employee with id, name, salary (union), type.
4. Input n employees.
5. Convert weekly salary to monthly and find maximum.
6. Display employee with maximum salary.
7. Stop

20. Program to combine structure array and union to store multiple employee types

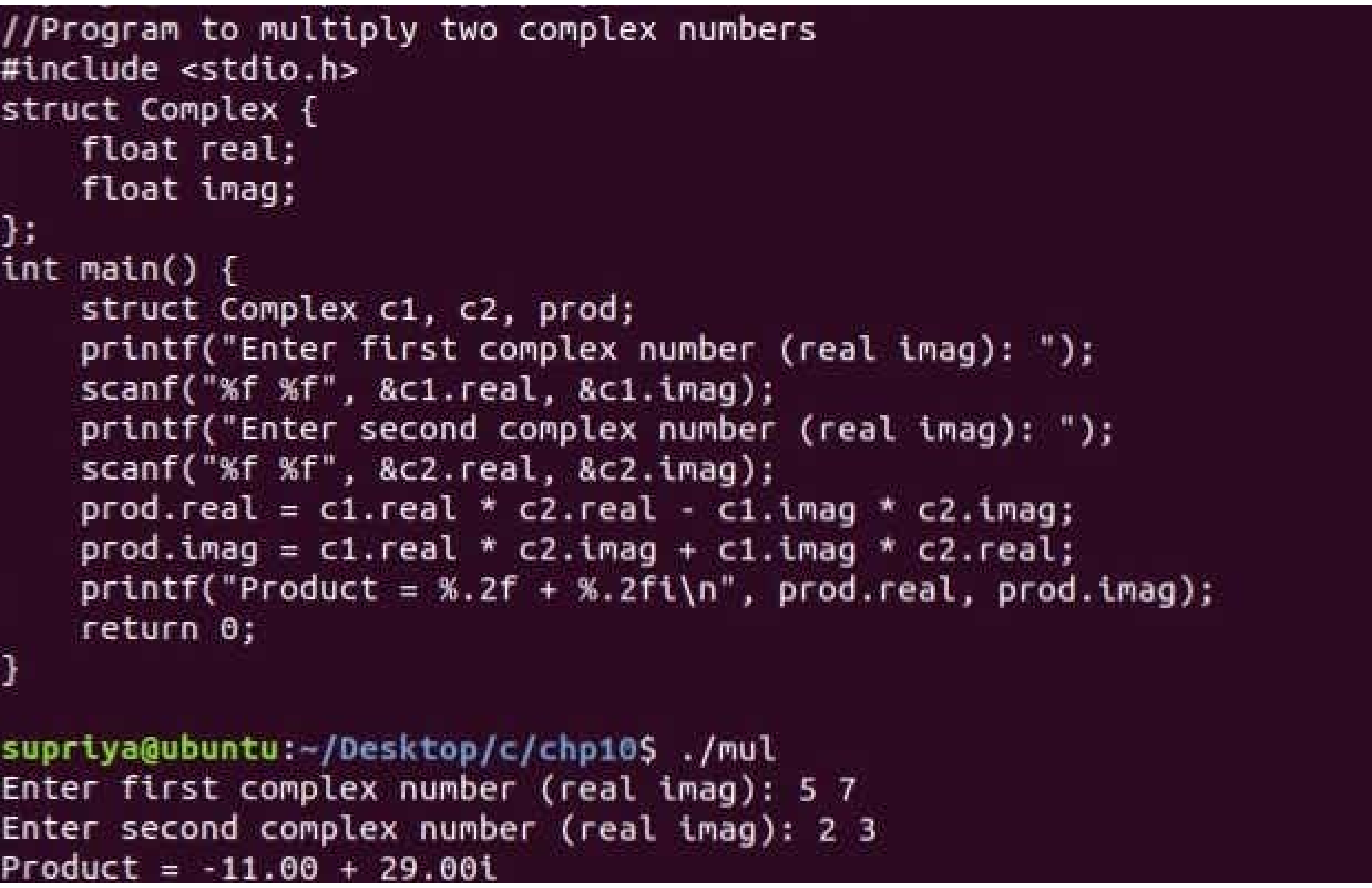
Algorithm

1. Start
2. Define union Salary with monthly or weekly.
3. Define structure Employee with id, name, salary (union), type.
4. Input n employees.
5. Display all employee details, converting weekly → monthly if needed.
6. Stop



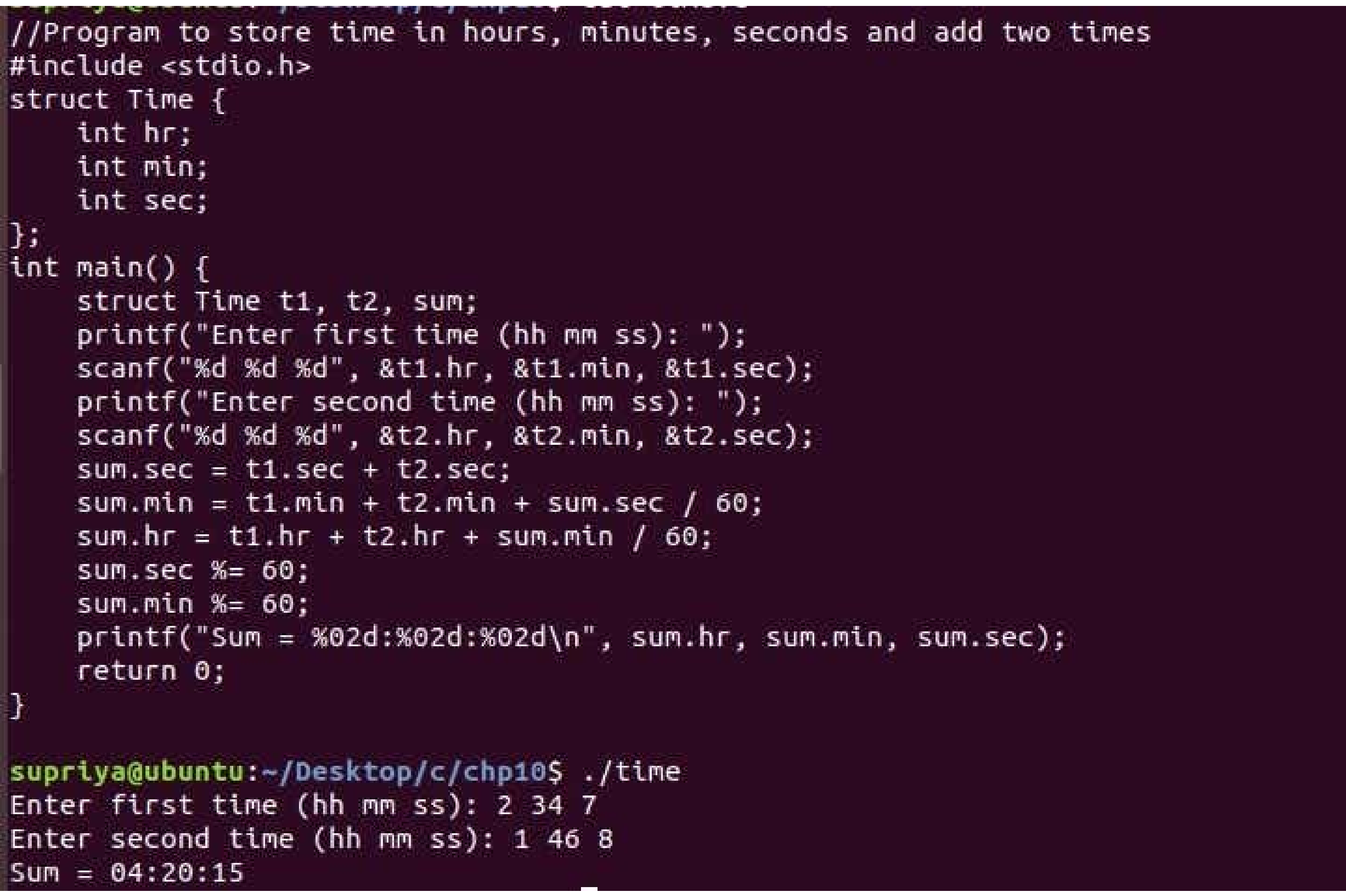
```c
//Program to add two complex numbers using structures
#include <stdio.h>
struct Complex {
    float real;
    float imag;
};
int main() {
    struct Complex c1, c2, sum;
    printf("Enter first complex number (real imag): ");
    scanf("%f %f", &c1.real, &c1.imag);
    printf("Enter second complex number (real imag): ");
    scanf("%f %f", &c2.real, &c2.imag);
    sum.real = c1.real + c2.real;
    sum.imag = c1.imag + c2.imag;
    printf("Sum = %.2f + %.2fi\n", sum.real, sum.imag);
    return 0;
}
```

```
supriya@ubuntu:~/Desktop/c/chp10$ ./com
Enter first complex number (real imag): 4 9
Enter second complex number (real imag): 2 3
Sum = 6.00 + 12.00i
```



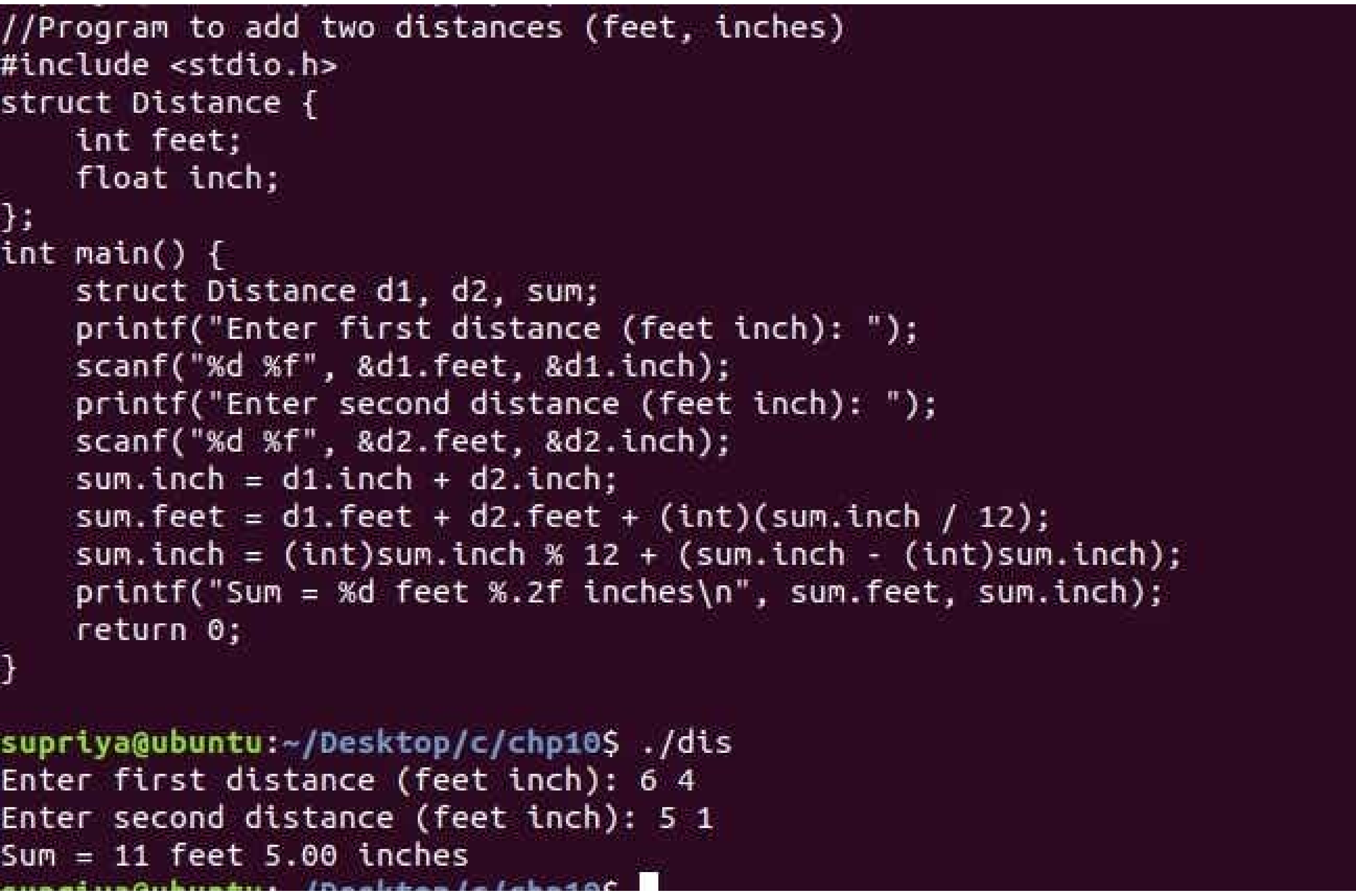
```
//Program to multiply two complex numbers
#include <stdio.h>
struct Complex {
    float real;
    float imag;
};
int main() {
    struct Complex c1, c2, prod;
    printf("Enter first complex number (real imag): ");
    scanf("%f %f", &c1.real, &c1.imag);
    printf("Enter second complex number (real imag): ");
    scanf("%f %f", &c2.real, &c2.imag);
    prod.real = c1.real * c2.real - c1.imag * c2.imag;
    prod.imag = c1.real * c2.imag + c1.imag * c2.real;
    printf("Product = %.2f + %.2fi\n", prod.real, prod.imag);
    return 0;
}

supriya@ubuntu:~/Desktop/c/chp10$ ./mul
Enter first complex number (real imag): 5 7
Enter second complex number (real imag): 2 3
Product = -11.00 + 29.00i
```



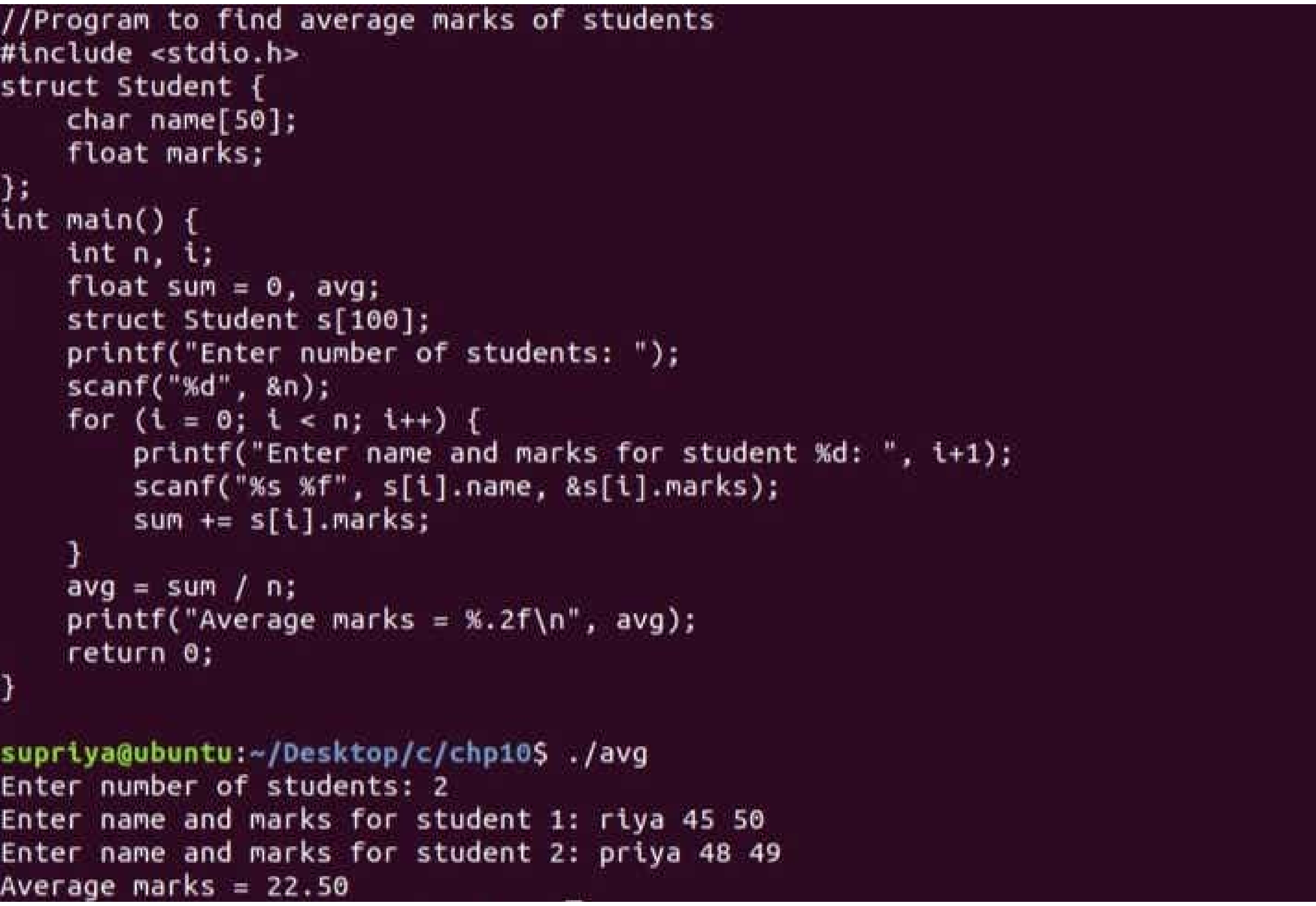
```
//Program to store time in hours, minutes, seconds and add two times
#include <stdio.h>
struct Time {
    int hr;
    int min;
    int sec;
};
int main() {
    struct Time t1, t2, sum;
    printf("Enter first time (hh mm ss): ");
    scanf("%d %d %d", &t1.hr, &t1.min, &t1.sec);
    printf("Enter second time (hh mm ss): ");
    scanf("%d %d %d", &t2.hr, &t2.min, &t2.sec);
    sum.sec = t1.sec + t2.sec;
    sum.min = t1.min + t2.min + sum.sec / 60;
    sum.hr = t1.hr + t2.hr + sum.min / 60;
    sum.sec %= 60;
    sum.min %= 60;
    printf("Sum = %02d:%02d:%02d\n", sum.hr, sum.min, sum.sec);
    return 0;
}

supriya@ubuntu:~/Desktop/c/chp10$ ./time
Enter first time (hh mm ss): 2 34 7
Enter second time (hh mm ss): 1 46 8
Sum = 04:20:15
```



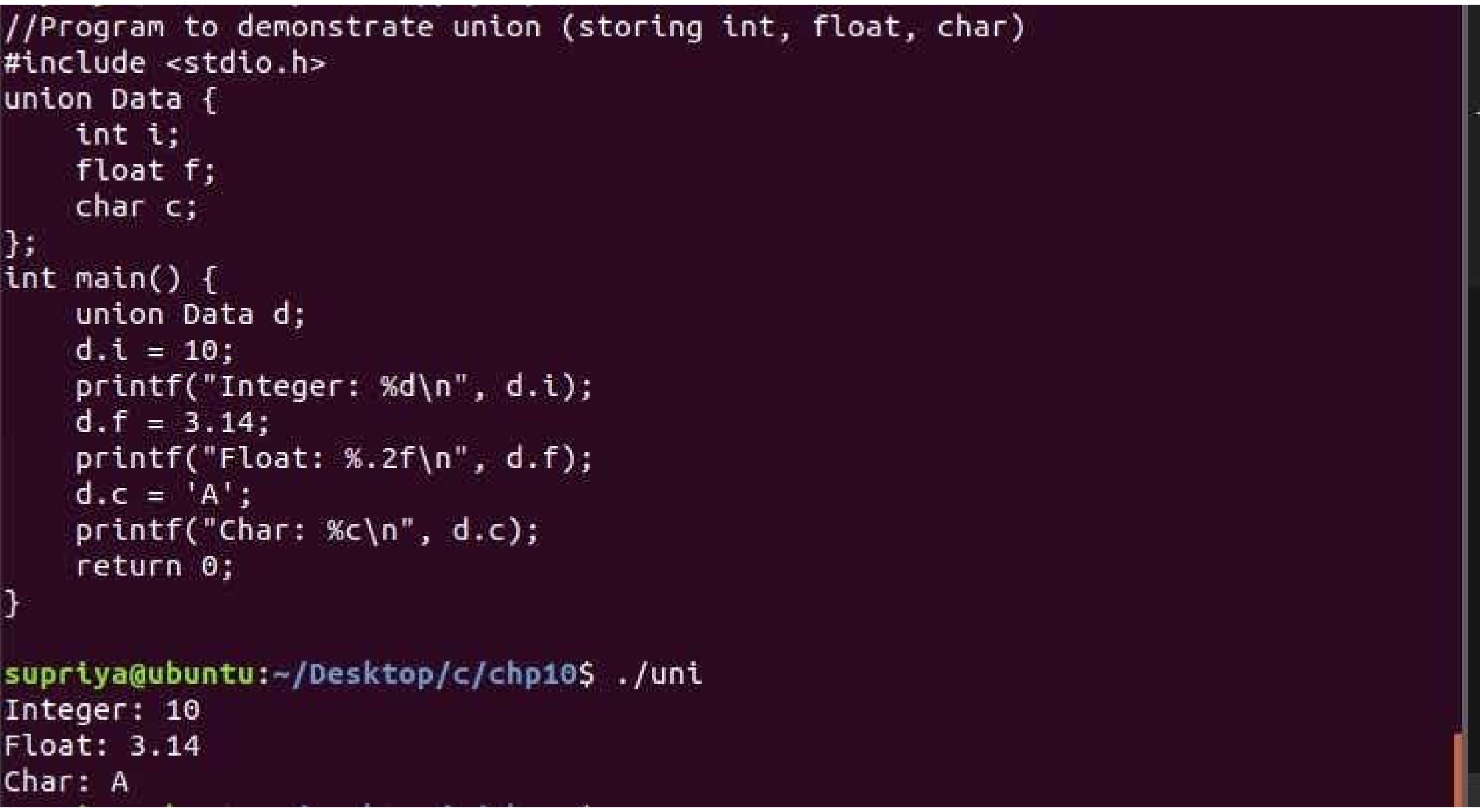
```c
//Program to add two distances (feet, inches)
#include <stdio.h>
struct Distance {
    int feet;
    float inch;
};
int main() {
    struct Distance d1, d2, sum;
    printf("Enter first distance (feet inch): ");
    scanf("%d %f", &d1.feet, &d1.inch);
    printf("Enter second distance (feet inch): ");
    scanf("%d %f", &d2.feet, &d2.inch);
    sum.inch = d1.inch + d2.inch;
    sum.feet = d1.feet + d2.feet + (int)(sum.inch / 12);
    sum.inch = (int)sum.inch % 12 + (sum.inch - (int)sum.inch);
    printf("Sum = %d feet %.2f inches\n", sum.feet, sum.inch);
    return 0;
}
```

```
supriya@ubuntu:~/Desktop/c/chp10$ ./dis
Enter first distance (feet inch): 6 4
Enter second distance (feet inch): 5 1
Sum = 11 feet 5.00 inches
```



```
//Program to find average marks of students
#include <stdio.h>
struct Student {
    char name[50];
    float marks;
};
int main() {
    int n, i;
    float sum = 0, avg;
    struct Student s[100];
    printf("Enter number of students: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        printf("Enter name and marks for student %d: ", i+1);
        scanf("%s %f", s[i].name, &s[i].marks);
        sum += s[i].marks;
    }
    avg = sum / n;
    printf("Average marks = %.2f\n", avg);
    return 0;
}
```

```
supriya@ubuntu:~/Desktop/c/chp10$ ./avg
Enter number of students: 2
Enter name and marks for student 1: riya 45 50
Enter name and marks for student 2: priya 48 49
Average marks = 22.50
```



```
//Program to demonstrate union (storing int, float, char)
#include <stdio.h>
union Data {
    int i;
    float f;
    char c;
};
int main() {
    union Data d;
    d.i = 10;
    printf("Integer: %d\n", d.i);
    d.f = 3.14;
    printf("Float: %.2f\n", d.f);
    d.c = 'A';
    printf("Char: %c\n", d.c);
    return 0;
}

supriya@ubuntu:~/Desktop/c/chp10$ ./uni
Integer: 10
Float: 3.14
Char: A
```