COMP133 –COMPUTER AND PROGRAMMING

Structures

Dr. Radi Jarrar
Department of Computer Science
Birzeit University

* The material in this lecture notes is adapted from the slides of Mr. Abdallah Karakra, Birzeit University, 2016

User-Defined Structure Types

- A database is a collection of information subdivided into records.
- A record is a collection of information of one data object (e.g., ID, name, and age of a student).
- C allows us to define a new data type (called structure type) for each category of a structured data object.

User-Defined Structure Types

- A structure is a collection of related data items, possibly of different types.
- A struct is heterogeneous in that it can be composed of data of different types.
- Array is homogeneous since it can contain only data of the same type.

Declaring Structure Types

- Syntax

```
typedef struct{type1 id1;
               type2 id2;…
          } struct_type;
```

- Example:

```
typedef struct{char name [20];
               int StudentID;
               } student_info;
```

Declaring Structure Types

- Declaration:

```
student_info student1, student2 = {“Hisham”, 119999};
```

- Accessing Members of a struct:

- `student1.name` **- is the name of student**
- `student1.age` **- is the ID of student**

- **Members of a structtype variable are accessed with the dot (.) operator**

Example

```
typedef struct{
      char name [20];
      int age;
} student_info;
```

Declare variable

- `student_info student1;`
- `strcpy(student1.name, "Alma");`
- `student1.age = 9;`

student_info								
name	A	l	m	a	'\o'			...
age	9							

Example

Without using typedef:

```
struct student_info{
      char name [20];
      int age;
};
```

Declare variable

- `struct student_info student1;`
- `strcpy(student1.name, “Alma");`
- `student1.age = 9;`

Example

```
typedef struct{
        char name [20];
        intage;
    } student_info;
```

- **//Declare variable**
- `student_info student2;`
- **//Assigning values to student2 (from a user)**
- `scanf("%s%d", student2.name, &student2.age);`
- **//Printing the values of student2**
- `printf("%s%d", student2 .name, student2.age);`

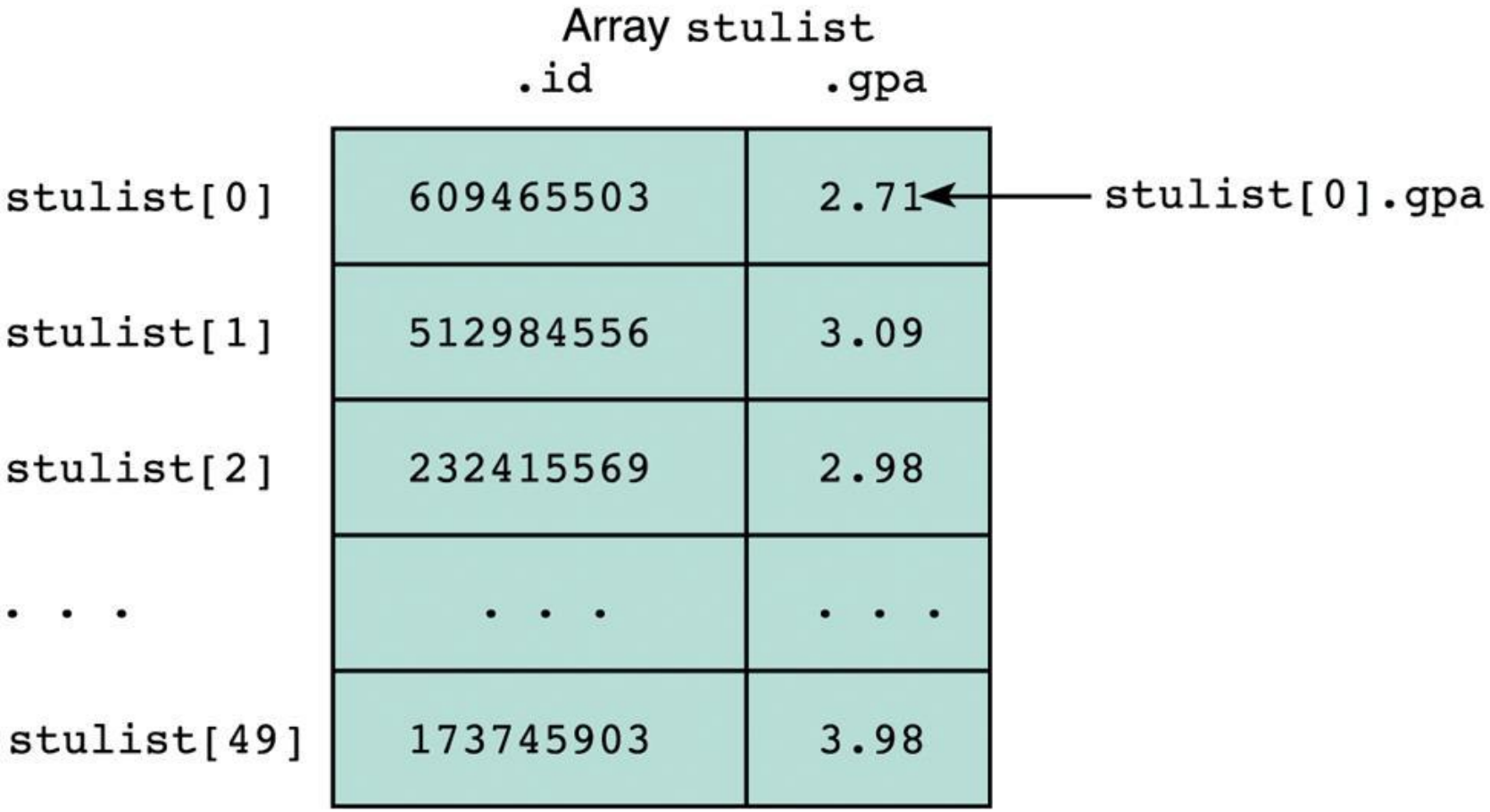
Creating Array of Structs

- Create an array of 50 student_info structures
-
```
typedef struct{
        int id;
        double gpa;
    } student_t;
```

- Usage:

```
student_t stulist[50];
stulist[3].id = 1202023;
stulist[3].gpa= 3.0;
```

Creating Array of Structs



Example – Array of structs

- A C Program to Store Information(name, id, and grade) of a Student Using Structure.

```c
#include <stdlib.h>
typedef struct {
 char names[20];
 int id;
 int grade;

} Student_t;
int main()
{    int i;
    Student_t info[5]; //array of students
    printf ("Please enter student information: name,id and grade: \n");
    //Fill array
    for (i=0;i<5;i++)
        scanf ("%s %d %d",info[i].names,&info[i].id,&info[i].grade);

    // Print array
    for (i=0;i<5;i++)
        printf ("\n%s %d %d",info[i].names,info[i].id,info[i].grade);
    return 0;
}
```

Example – Array of structs

```
Please enter student information: name,id and grade:
Yamen 100228   99
Sandy 101000   98
Amer  107342   90
Amera 100982   93
Lina  100988   95

Yamen 100228 99
Sandy 101000 98
Amer 107342 90
Amera 100982 93
Lina 100988 95
```

Example

- C Program to Store Information (name, roll and marks) of a Student Using Structure.

```c
#include <stdio.h>
typedef struct {
    char name[50];
    int roll;
    float marks;
} student_t;
int main(){
    student_t s;
    printf("Enter information of students:\n\n");
    printf("Enter name: ");
    scanf("%s",s.name);
    printf("Enter roll number: ");
    scanf("%d",&s.roll);
    printf("Enter marks: ");
    scanf("%f",&s.marks);
    printf("\nDisplaying Information\n");
    printf("Name: %s\n",s.name);
    printf("Roll: %d\n",s.roll);
    printf("Marks: %.2f\n",s.marks);
    return 0;
}
```

Example - Run

Output

```
Enter information of students:

Enter name: Adele
Enter roll number: 21
Enter marks: 334.5

Displaying Information
name: Adele
Roll: 21
Marks: 334.50
```

Passing A Structure to a Function

- Example, a C program that adds two complex numbers by passing structure to a function.
- A complex number is a number of a real part and an imaginary part (a+bi)

```c
#include <stdio.h>
typedef struct{
    float real;
    float imag;
}complex_t;
complex_t add(complex_t n1,complex_t n2);
int main(){
    complex_t n1,n2,temp;
    printf("For 1st complex number \n");
    printf("Enter real and imaginary respectively:\n");
    scanf("%f%f",&n1.real,&n1.imag);
    printf("\nFor 2nd complex number \n");
    printf("Enter real and imaginary respectively:\n");
    scanf("%f%f",&n2.real,&n2.imag);
    temp=add(n1,n2);
    printf("Sum=%.1f+%.1fi",temp.real,temp.imag);
    return 0;
}
complex_t add(complex_t n1,complex_t n2){
      complex_t temp;
      temp.real=n1.real+n2.real;
      temp.imag=n1.imag+n2.imag;
      return(temp);
}
```

Passing A Structure to a Function

Output

```
For 1st complex number
Enter real and imaginary respectively: 2.3
4.5

For 1st complex number
Enter real and imaginary respectively: 3.4
5
Sum=5.7+9.5i
```

Example – Filling a struct from a function (output parameter)

```c
#include <stdio.h>
typedef struct{
 char name[20];
 int age;
}student_t;

void fillStruct(student_t*);
int main()
{
    student_t s1,s3;
    student_t *s2;
    s2=&s1;

    fillStruct(s2); //fillStruct(&s1);
    s3=s1;
    printf("%s %d",s3.name,s3.age);
    return 0;
}
void fillStruct(student_t*ptr)
{
    scanf("%s%d",(*ptr).name,&ptr->age);//  (*ptr).name same as ptr->name
}
```

Example – Filling a struct from a function (output parameter)

```c
#include <stdio.h>
typedef struct{
 char name[20];
 int age;
}student_t;

void fillStruct(student_t*);
int main()
{
    student_t s1,s3;
    student_t *s2;
    s2=&s1;

    fillStruct(s2); //fillStruct(&s1);
    s3=s1;
    printf("%s %d",s3.name,s3.age);
    return 0;
}
void fillStruct(student_t*ptr)
{
    scanf("%s%d",(*ptr).name,&ptr->age);//  (*ptr).name same as ptr->name
}
```