Exercise 10: Structures

S Milton Rajendram

2-6 April 2018 (Mon-Thurs)

1 Arrays of structures (1)

1. Represent an employee by a structure, having these members:
   (a) employee code
   (b) name
   (c) designation
   (d) pay: basic pay, hra, da
2. Employees of a company can be represented by an array of structures. Write a function to populate the employees array.
3. Write a function to print the salary slip of a given employee.
4. Drive your functions from `main()` and test them.

1.1 Specification

A structure `employees`, which gets the required elements, and 2 functions `populate()` and `print_salary()`, which gets an array of pointers to structures and its size as input, assigns the values and prints the values respectively.

1.2 Prototype

```
void populate(Employees* e[],int n);

void print_salary(Employees* e[],int n);
```

1.3 Program Design

The program consists of a structure, 2 functions `populate(Employees* e[],int n)`, `print_salary(Employees* e[],int n)`, which does the required actions, and `main()`, which gets input from `stdin`, and calls the functions.

1.4 Algorithm

```
def populate(e,n):
  for i in range(n):
    //get the input from the user
    e[i].code=code
    e[i].name=name
    e[i].designation=designation
    e[i].bp,e[i].hra,e[i].da=bp,hra,da
def print_salary(e,n):
  // get code from user
  for i in range(n):
    if e[i].code==code:
      //print bp,hra,da
```

1.5 Source Code

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define N 1000
struct employees{
  int code;
  char name[100];
  char designation[100];
  int bp,hra,da;
};
typedef struct employees Employees;
void populate(Employees* e[],int n){
  int code,i;
  char name[100];
  char designation[100];
  int bp,hra,da;
  for(i=0;i<n;i++){
    scanf("%d%s\n%s\n%d%d%d",&code,name,designation,&bp,&hra,&da);
    e[i]=(Employees*)malloc(N);
    e[i]->code=code;
    strcpy(e[i]->name,name);
    strcpy(e[i]->designation,designation);
    e[i]->bp=bp;
    e[i]->hra=hra;
    e[i]->da=da;
  }
}
void print_salary(Employees* e[],int n){
  int i,t;
  scanf("%d",&t);
  for(i=0;i<n;i++){
    if(e[i]->code==t)
      printf("%d\n%d\n%d\n",e[i]->bp,e[i]->hra,e[i]->da);
  }
}

int main(){
  int n;
  Employees* e[100];
  scanf("%d",&n);
  populate(e,n);
  print_salary(e,n);
}
```

1.6 Test Input

```
2
```

```
1 Ram

CEO

50000 25000 10000

2 RSB

COO

25000 25000 10000

1
```

1.7 Output

```
50000

25000

10000
```

2 Arrays of structures (2)

1. Define a structure to represent a student. It should store 3 UT marks for a subject and the final internal mark for that subject.

   ```
   struct student {
      int  rollnum;
      char name[100];
      int  ut[4];
   };
   ```

   Write functions to create a student structure and initialzie it. Write a function to print a student struture.

2. Read the roll numbers, names and ut marks for 3 unit tests for 10 students from stdin.

3. Write a function to calculate the final internal mark for each student.

4. Modify the structure to store the ut marks and internal mark in 5 subjects. Write a function to compute internal marks of students for 5 different subjects.

2.1 Specification

A structure `student`, which consists the necessary elements, and 2 functions `populate()` and `print()`, which gets an array of pointers to structures and its size as input, assigns the values and prints the values respectively.

2.2 Prototype

```
void populate(Student* s[],int n);

void print_salary(Student* s[],int n);
```

2.3 Program Design

The program consists of a structure, 2 functions `populate(Student* s[],int n)`, `print_salary(Student* s[],int n)`, which does the required actions, and `main()`, which gets input from `stdin`, and calls the functions.

2.4 Algorithm

```
def populate(s,n):
  for i in range(n):
    //get the input from the user
    s[i].rollnum=rnum
    s[i].name=q
    s[i].ut[0],s[i].ut[1],s[i].ut[2]=a,b,c
    r=a+b+c
    s[i].ut[3]=r/15
def print_salary(s,n):
  for i in range(n):
      print(s[i]->ut[3]
```

2.5 Source Code

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define N 1000
struct student{
  int rollnum;
  char name[100];
  int ut[10];
```

```
};
typedef struct student Student;
void populate(Student* s[],int n){
  int rnum;
  char q[100];
  int a,b,c;
  for(int i=0;i<n;i++){
    int r=0;
    s[i]=(Student*)malloc(N);
    scanf("%d%s\n%d%d%d",&rnum,q,&a,&b,&c);
    s[i]->rollnum=rnum;
    strcpy(s[i]->name,q);
    s[i]->ut[0]=a;
    s[i]->ut[1]=b;
    s[i]->ut[2]=c;
    for(int j=0;j<3;j++){
      r+=s[i]->ut[j];
    }
    s[i]->ut[3]=r/15;
  }
}
void print(Student* s[],int n){
  for(int i=0;i<n;i++){
    printf("%d\n",s[i]->ut[3]);
  }
}
int main(){
  int n;
  Student* s[100];
  scanf("%d",&n);
  populate(s,n);
  print(s,n);
}
```

2.6 Test Input

```
3
125 Ram
100 96 100
114 Praveen
100 100 92
93 Nanda
95 100 100
```

2.7 Output

```
19
19
19
```

3 Number conversion

Convert the given decimal number into binary, octal and hexadecimal numbers using user defined functions.

3.1 Specification

3 functions `binary()`, which takes the number and an integer array as input, finds the number in binary, stores it in the array and returns the length, `octal()`, which takes the number and an integer array as input, finds the number in octal, stores it in the array and returns the length, and `hexadecimal()`, which takes the number and character array as input, finds the number in hexadecimal, stores it in the array and returns the length.

3.2 Prototype

```
int binary( int n, int a[]);
int octal( int n, int a[]);
int hexadecimal(int n, char a[]);
```

3.3 Program Design

The program consists of 3 functions `binary( int n, int a[])`,`octal( int n, int a[])`,`hexadecimal(int n, char a[])`, which convert the given number into the required form, and `main()`, which gets the input from `stdin`, calls the functions and prints the result on `stdout`.

3.4 Algorithm

```
def binary(n,a):
  static k=0 //k holds its value
  if n==1:
    a[k++]=n
  else:
    int b=n%2
    binary(n/2,a)
    a[k++]=b
  return k
def octal(n,a):
  static k=0
  if n<8:
    a[k++]=n
  else:
    int b=n%8
    octal(n/8,a)
    a[k++]=b
  return k
def hexadecimal(n,a):
  static k=0
  if n<10:
    a[k++]=48+n
  elif 10<=n<16:
    a[k++]=55+n
  else:
    int b=n%16
    hexadecimal(n/16,a)
    if b<10:
      a[k++]=48+b
    else:
      a[k++]=55+b
  return k
```

3.5 Source Code

```
#include<stdio.h>
#include<string.h>
#define MAX 1000
int binary( int n, int a[]){
  static int k=0;
  if(n==1){
    a[k++]=n;
  }
  else{
    int a=n%2;
    binary(n/2,a);
    a[k++]=a;
  }
  return k;
}
int octal( int n, int a[]){
  static int b=0;
  if(n<8){
    a[b++]=n;
  }
  else{
    int a=n%8;
    binary(n/8,a);
    a[b++]=a;
  }
  return b;
}
int hexadecimal(int n, char a[]){
  static int p=0;
  if(n<10){
    a[p++]=48+n;
  }
  else if(10<=n && n<16){
    a[p++]=55+n;
  }
  else{
    int b=n%16;
    hexadecimal(n/16,a);
    if(b<10){
      a[p++]=48+b;
    }
    else if(10<=b && b<16){
      a[p++]=55+b;
    }
  }
  return p;
}
}
int main(){
  int n;
  int bi[MAX],oc[MAX];
  char he[MAX];
  scanf("%d",&n);
  int d=binary(n,bi);
  for(int i=0;i<d;i++){
    printf("%d",bi[i]);
  }
  printf("\n");
  int q=octal(n,oc);
  for(int i=0;i<q;i++){
    printf("%d",oc[i]);
  }
  printf("\n");
  int r=hexadecimal(n,he);
  for(int i=0;i<r;i++){
    printf("%c",he[i]);
  }
}
```

3.6 Test Input

```
43
```

3.7 Output

```
101011
53
2B
```