CHAPTER - 13

STRUCTURES

CHAPTER 13

STRUCTURES

STRUCTURES

DEFINING A STRUCTURE

- A structure is a data type containing multiple members defined together which is defined like a data type of structure.
- Contiguous memory is allocated for a structure variable.
- A variable name structure of defined data type is declared.

format of structure definition:

```
struct optional_name {
    data_type1  variable_name;
    data_type2  variable_name;
    data_type3  variable_name;
    data_type4  variable_name;
} struct_name;
```

example of structure:

```
struct  {
        char name[64];
        char course[128];
        int age;
        int year;
} student;
student  st_rec;
student * st_ptr;
```

Accessing members of structure:

```
 st_rec.name  /* dot operator */
st_ptr -> name /* pointer use */
```

STRUCTURES

TYPE DEFINITIONS

format:

```
typedef prev_declaration curr_declaration;

#include<stdio.h>

typedef int *intptr;
int main ()
{
        intptr ip;
        return 0;
}
```

Type definitions for a structure:

```
#include<stdio.h>
typedef struct person
{
        char *name;
        int age;
} PERSON;
int main ()
{
        PERSON p;
        p.name = "John Smith";
        p.age = 25;
        printf ("%s", p.name);
        printf ("%d", p.age);
        return 0;
}
```

STRUCTURES

POINTERS AND STRUCTURES

```
struct products
{
     char name[30];
     int manufac;
}
float net;
struct products item[2], *ptr;
ptr = item;
ptr- >name;
ptr- >manufac;
ptr- >net;
(*ptr).manufac = 75;
for (ptr = item;  ptr <  item + 2; ptr++)
     printf (“%s%d%f\n”, ptr- >name,
               ptr->manufac, ptr- >net);
```

Type definitions for a structure:

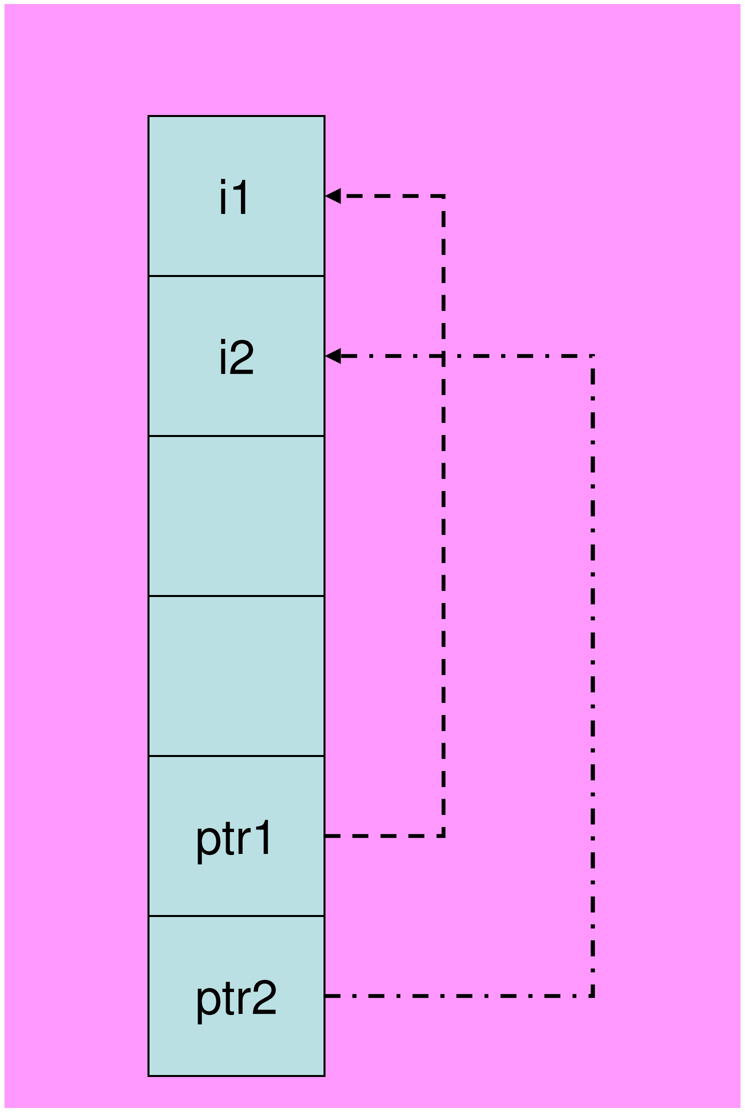
```
#include<stdio.h>
typedef struct person
{
        char *name;
        int age;
} PERSON;

int main ()
{
        PERSON p;
        p.name = "John Smith";
        p.age = 25;
        printf ("%s", p.name);
        printf ("%d", p.age);
        return 0;
}
```

STRUCTURES

STRUCTURES CONTAINIING POINTERS

```
#include <stdio.h>
void main()  /* structures containing pointers */
{
      struct  int_pointers {
                int  *ptr1, *ptr2;
      };
      struct int_pointers  ptrs;
      int  i1 = 154, i2;
      ptrs.ptr1 =  &i1;
      ptrs.ptr2 =  &i2;
      *ptrs.ptr2 =  -97;
      printf ("i1 = %d, *ptrs.ptr1 = %d\n", i1,
                                        *ptrs.ptr1);
      printf ("i2 = %d, *ptrs.ptr2 = %d\n", i2,
                                        *ptrs.ptr2)
}
```



STRUCTURES

RANDOM NUMBER GENERATION

The rand function computes a sequence of pseudo-random integers in the range of 0 to RAND_MAX (a symbolic constant defined in the <stdlib.h> header file). The rand function returns a pseudo-random integer.

The ANSI standard states that the value of RAND_MAX must be at least 32767, which is the maximum value for a two-byte (16 bit) integer. If rand truly produces integers at random, every number between 0 and RAND_MAX has an equal chance (or probability) of being chosen each time rand is called.

A dice-rolling program that simulates a six-sided die would require random integers in the range of 1 to 6. The modulus operator (%) can be used in conjunction with the function rand to produce integers of random numbers with in the desired range.

STRUCTURES

RANDOM NUMBER GENERATION

This is called scaling. The number produced by the above combination would be within the range of 0 to 5. Adding 1 can shift the range of numbers to the previous result to produce the values ranging between 1 and 6. The values produced n using rand function, shifting value a, and scaling factor b can be generalized as:

n = a + rand () % b;

The rand function actually generates pseudo-random numbers. Important characteristic of the rand function is its repeatability. Calling rand function repeatedly produces a sequence of numbers that appears to be random.

Examples of rand and srand:

```
#include  <stdio.h>
#include  <stdlib.h>
#define  DICE     6
#define  SHIFT     1
#define  PSEUDO  20
int main()   // sorts an values in ascending
{
      for ( int i = 1; i <= PSEUDO; i++)
      {
                printf (“%15d”, SHIFT  + (rand ()
                                          % DICE  ) );
                if ( i % 5 == 0)
                printf (“\n);
      }
      return 0;
 }  /** end of main function   */
```

STRUCTURES

RANDOM NUMBER GENERATION

output of the previous program:

```
   5        5        3        5        5
   2        4        2        5        5
   5        3        2        2        1
   5        1        4        6        4
```

←- - - - - - - - - - - - - - - - - - - -→

output of the next program:

```
Please enter the seed: 67
   1        6        5        1        4
   5        6        3        1        2
Please enter the seed: 432
   4        2        5        4        3
   2        5        1        4        1
Please enter the seed: 67
   1        6        5        1        4
   5        6        3        1        2
```

```
#include  <stdio.h>
#include  <stdlib.h>
#define  DICE      6
#define  SHIFT     1
#define  PSEUDO  10
int main()
{
      unsigned seed;
      printf ("Please enter the seed: ");
      scanf ("%u",  &seed);
      srand (seed);
      for ( int i = 1; i <= PSEUDO; i++)
      {
            printf ("%15d", SHIFT  + (rand ()
                                   % DICE  ) );
            if ( i % 5 == 0)  printf ("\n");
      }
      return 0;
•     }  /** end of main function   */
```

STRUCTURES

A CARDS GAME

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SUITSIZE  13
#define DECKSIZE  52
#define NUMSUITS   4
struct card {
     char *face;
     char *suit;
};
typedef struct card CARD;
void fillDeck (CARD *, char *[],
     char *[]);
void shuffle (CARD *);
void deal (CARD *);
```

```
void main ()
{
    char*suit [NUMSUITS] = {"Hearts",
     "Diamonds", "Clubs", "Spades"};
     char *face [SUITSIZE]  = {"Ace",
     “Deuce",      "Three", "Four", "Five",
     "Six", "Seven",  "Eight", "Nine", "Ten",
     "Jack", "Queen",  "King"};
     CARD  deck [DECKSIZE];
     srand (time (NULL) );
     fillDeck (deck, face, suit);
     shuffle (deck);
     deal (deck);
     return;
}  /** end of main function  */
```

STRUCTURES

A CARDS GAME

```
void fillDeck (CARD *wDeck[], char
      *wFace[], char *wSuit[])
{
      for (int i = 0; i <= DECKSIZE; i++)
      {
            wDeck[i].face = wFace [i %
                        SUITSIZE];
            wDeck[i].suit = wSuit[i /
                        SUITSIZE];
      }
      return;
}  /** end of function shuffle  */
```

```
void shuffle (CARD *wDeck)
{
      int i, j;
      CARD temp;
      for (i = 0;  i <= DECKSIZE;  i++)
      {
            j = rand () % DECKSIZE;
            temp        =  wDeck[i];
            wDeck[i]  =  wDeck[j];
            wDeck[j]  =  temp;
      }
      return;
}  /** end of function shuffle  */
```

STRUCTURES

A CARDS GAME

```
void deal (CARD *wDeck)
{
    printf("\n***********************************************\n");
    printf ("  Player 1        Player 2        Player 3            Player 4");
    printf("\n***********************************************\n");
    printf (" %5s, %-4s     %5s, %-4s   ", "Face", "Suit", "Face", "Suit");
    printf ("\n------------------------------------------------------------\n");
    for (int i = 0; i <= DECKSIZE; i++)
    {
            printf ("%5s, %-8s%c", wDeck[i].face, wDeck[i].suit,
            (i + 1) % 4 == 0 ? '\n' : '\t');
    }
    return;
}   /** end of function deal  */
```

STRUCTURES

A CARDS GAME

Output of the program:

```
*************************************************************************
  Player 1          Player 2          Player 3          Player 4
*************************************************************************
  Face, Suit        Face, Suit        Face, Suit        Face, Suit
---------------------------------------------------------------------------
 Five,  Clubs       Deuce, Hearts     Ace, Diamonds     Six, Diamonds
Six,  Hearts        Queen, Clubs      Eight, Hearts     King, Clubs
Jack,  Hearts       Ten, Hearts       Deuce, Spades     Eight, Diamonds
Three, Clubs        Six, Clubs        Ace, Clubs        Ten, Clubs
Queen, Diamonds    Seven, Spades      Three, Diamonds   Queen, Hearts
Nine, Clubs         Seven, Clubs      Ten, Diamonds     Queen, Spades
Deuce, Clubs        Nine, Spades      Deuce, Diamonds    Nine, Hearts
Four, Hearts        Four, Clubs       Nine, Diamonds     Jack, Diamonds
Eight, Spades       Five, Spades      Five, Hearts       King, Spades
Five, Diamonds      Ace, Spades       Jack, Spades       Six, Spades
King, Diamonds      King, Hearts      Three, Spades     Ace, Hearts
Jack, Clubs         Seven, Diamonds   Four, Diamonds    Eight, Clubs
Three, Hearts       Four, Spades      Ten, Spades       Seven, Hearts
```