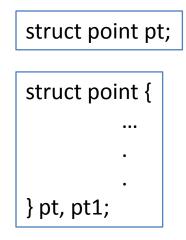
## 8. Structures, File I/O, Recursion

18<sup>th</sup> October IIT Kanpur

## **Basic of Structures**

 Definition: A collection of one or more different variables with the same handle (same name).

struct point {		
char name[30];		
int x;		
int y;		
double temperature;		
}		



## Basic of Structures contd...

• Access an element

structure-name.member

• Example

printf("x = %d, y = %d\n", pt.x, pt.y);

### {program: basic\_of\_structures.c}

## Basic of Structures contd...

• Structs can also contain other structs.

```
struct rectangle {
    struct point pt1;
    struct point pt2;
};
```

```
struct rectangle rect;
```

• To access its element:

#### rect.pt1.x;

## **Structures and Functions**

- When structures are passed into functions all of their values are copied. (pass by value)
- A function must return the structure to affect the target structure.

{program: structures\_and\_functions.c}

{program: structures\_and\_functions1.c}

- This is a lot of copying of variable values onto and off the stack. (inefficient)
- Pointers will be used to make this better.

## Arrays of Structures

• Array of Structures act like any other array.

struct point pt[3];

pt[0].name = "A"; pt[0].x = 0; pt[0].y = 1; pt[1].name = "B"; pt[1].x = 4; pt[1].y = 1; pt[2].name = "mid"; pt[2].x = (pt[0].x + pt[1].x)/2; pt[2].y = (pt[0].y + pt[1].y)/2;

Memory occupied: the dimensions of the array multiply by sizeof(struct tag)
 – (Remember) sizeof() is compile time function

## Pointers to Structures

- Pointers are an easier way to manipulate structure members by reference
- The entire structure is not passed by value, only the address of the first member
- Use arrow operator for accessing the struct element

```
struct Date MyDate, *DatePtr;
DatePtr = &MyDate;
DatePtr->month = 2;
DatePtr->day = 22;
```

## Pointer to Structures contd...

• Example

```
struct Date {
    int month;
    int day;
    int year;
};
void AddDecade(struct Date *tmp) {
    tmp->year += 10; // or (*tmp).year += 10;
}
```

### {program: structures\_and\_functions\_wPtr.c}

## Self referencing Structures

- Useful in data structures like trees, linked lists.
- It is illegal for a structure to contain an instance of itself.
  - Soln: Have a pointer to another instance.

```
struct tnode { /* the tree node */
    char *word;
    int count;
    struct tnode *left; /* left child */
    struct tnode *right; /* right child */
};
```

# Typedef

• Use typedef for creating new data type names

typedef int length;

this the name length a synonym for int. Afterwards, you can do:

length number = 4;

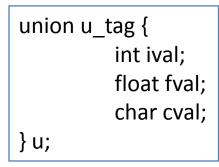
• In context of structs, you can do:

typedef struct tnode \*TreePtr;

typedef struct tnode {
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
•
•
} TreeNode;
,

## Unions

• A union is a memory location that is shared by two or more different types of variables.



- Each of ival, fval, cval have the same location in memory.
- Usage is similar to that of structs:

u.ival or u.cval

## Bit-fields

• When storage is high cost affair, we need to use memory efficiently (e.g in embedded systems)

struct {

unsigned pin1 : 1; unsigned pin2 : 1; unsigned pin3 : 1; } flags;

- Here each of the element takes a bit of memory (1 bit)
- The number following the colons represent the field length in bits.

# FILE I/O

• The file pointer

FILE \*fp;

• Opening a file

FILE \*fp = fopen("data.txt", "r");

- Modes
  - r : read, w: write, a: append, r+ : read and create if file does not exist, w+, a+, rb, wb, ab, r+b, r+w, r+a
- Closing a file



# FILE I/O contd...

fopen()	opens a file
fclose()	closes a file
fputc()	writes a character to a file
fgetc()	reads a character from a file
fputs()	writes a string to a file
fgets()	reads a string to a file
fseek()	change file position indicator
ftell()	returns to file position indicator
fprintf()	similar to printf(), but to a file instead of console
fscanf()	similar to scanf(), but to a file instead of console
remove()	deletes the file
fflush()	flushes the file pipe

#### Some functions for file I/O

## Supplement topic – I/O from console

- Reading from console
- During program execution
   printf(), scanf(), putc(), getc()
- Just before execution starts (parameters passed to the program)

\$ ./a.out 3 santa\_singh banta\_singh happy\_singh

int main(int argc, char \*argv[])

- argc: number of arguments (in above case, 5)
- argv: pointer to array of char pointers

## More supplement - Recursion

- Recursion is when a function calls itself.
  - Great Utility
  - Makes the code easier
- Requirements to use recursion
  - A condition to cease at
    - otherwise the program would never terminate
    - the condition is usually written at the beginning of the recursive method

## Recursion contd...

#### • example:

```
/* non-recursive */
int fact(int n) {
    int t, answer;
    answer = 1;
    for(t=1; t<=n; t++)
    answer=answer*(t);
    return(answer);
}</pre>
```

```
/* recursive */
int factr(int n) {
    int answer;
    if(n==1) return(1);
    answer = factr(n-1)*n; /* recursive call */
    return(answer);
} C Course, Programming club, Fall 2008
```