Class 7

classnote

Pointers , Structures and Files

Class 07

Topic: Pointers, Structures, Unions, Files

Pointers

A pointer is a variable that stores the memory address of another variable.

Syntax: `data_type *pointer_name;`

Example:

```
int x = 10;
int *ptr = &x;
printf("%d", *ptr); // Output: 10
```

Pointer operations

You can increment or decrement a pointer to move through memory locations (mainly in arrays)

Example:

```
int arr[5] = {1,2,3,4,5};
int *p = arr;
p++;
printf("%d", *p); // Output: 2
```

Pointer to pointer

Example:

```
int x = 5;
int *p = &x;
int **pp = &p;
printf("%d", **pp); // Output: 5
```

Pointers and functions

Example:

```
void update(int *a)
{
    *a = *a + 5;
}
int main()
{
    int num = 10;
    update(&num);
    printf("%d", num); // Output: 15
}
```

Dynamic memory allocation

`malloc()` allocates memory of given size

Example:

```
int *ptr = (int*) malloc(sizeof(int));
*ptr = 20;
printf("%d", *ptr); // Output: 20
```

`calloc()` allocates memory for array and initializes to 0

Example:

```
int *arr = (int*) calloc(5, sizeof(int));
printf("%d", arr[2]); // Output: 0
```

`free()` deallocates memory

Example:

```
free(ptr);
free(arr);
```

Structures

Structure groups variables of different types

Example:

```
struct student
{
    int id;
    char name[20];
    float marks;
};
struct student s1;
s1.id = 101;
strcpy(s1.name, "Maria");
s1.marks = 95.5;
printf("%s %d %.1f", s1.name, s1.id, s1.marks); // Output: Maria 101 95.5
```

Pointer to structure

Example:

```
struct student *ptr = &s1;
printf("%s", ptr->name); // Output: Maria
```

Files

Opening a file:

```
FILE *fp;
fp = fopen("file.txt", "w");
```

Writing to file:

```
fprintf(fp, "Hello World");
```

Closing file:

```
fclose(fp);
```

Reading from file:

```
fp = fopen("file.txt", "r");
```