Union

# Union in C

* union is a user defined data type
* In union, all members share the same memory location
* Only one Member will be active at a time.

Suppose we are accessing one of the data member of union then

we cannot access other data member since we can access single

data member of union because each data member shares same memory

# Union definition

union <union\_name>

{

data\_type memeber1;

data\_type memeber2;

.......

.......

};

# Size of union

Size of a union is taken according the size of largest member in union

#include <stdio.h>

union test1 { int x; int y; };

union test2 { int x; char y; };

union test3 { int arr[10]; char y; };

int main() {

printf ("sizeof(test1) = %d, sizeof(test2) = %d,"

"sizeof(test3) = %d", sizeof(union test1),

sizeof(union test2), sizeof(union test3));

return 0;

}

Output: Copy

sizeof(test1) = 4, sizeof(test2) = 4,sizeof(test3) = 40

# Pointers to unions ?

we can have pointers to unions and can access members using arrow operator (->)

#include <stdio.h>

union test { int x; char y; };

int main()

{

union test p1;

p1.x = 65;

union test \*p2 = &p1; // p2 is a pointer to union p1

printf("%d %c", p2->x, p2->y); // Accessing union members using pointer

return 0;

}

Output:

65 A

# What are applications of union ?

useful where we want to use same memory for two or more members

example :

We want to implement a binary tree data structure where each leaf node has a double data value, while each internal node has pointers to two children, but no data

struct NODE {

struct NODE \*left;

struct NODE \*right;

double data;

};

structure requires 16 bytes, with half the bytes wasted for each type of node

if we declare a node as following, then we can save space.

struct NODE {

bool is\_leaf;

union {

struct {

struct NODE \*left;

struct NODE \*right;

} internal;

double data;

} info;

};