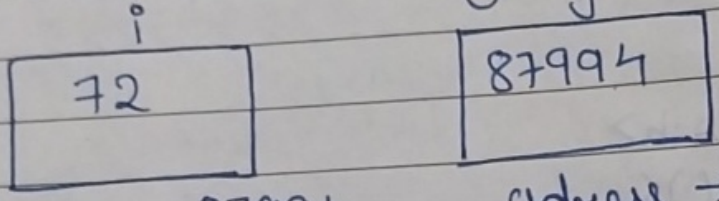
CHAPTER - 6 Pointers

A pointer is a variable which stores the address of another variable



j is a pointer
j points to i

The "address of" (&) Operator
The address of operator is used to obtain the address of a given variable

If you refer to the diagrams above
&i → 87994
&j → 87998

Format Specifier for printing pointer address is '%u'

The 'value at address' Operator (*)
The value at address or * operator is used to
Obtain the value present at a given memory address. It is denoted by *
* (&i) = 72
* (&j) = 87994

How to declare a pointer?

A pointer is declared using the following syntax

```
int * j;   ⇒ declare a variable j of
j = &i;        type int-pointer
   ||
Store address of i in j.
```

Just like pointer of type integer, we also have pointer of char, float etc.

```
int * ch→ptr;   → pointer to integer
char * ch→ptr;  → pointer to character
float * ch→ptr; → pointer to float
```

A program to demonstrate pointers

```
#include <stdio.h>
int main () {
    int i = 8;
    int * j;
    j = &i;
    printf("Add i = %u\n", &i);
    printf("Add i = %u\n" j);
    printf("Add j = %u\n", &j);
    printf("value i = %d\n", i);
    printf("value i = %d\n", *(&i));
    printf("value i = %d\n", *j);
    return 0;
}
```

Output :-

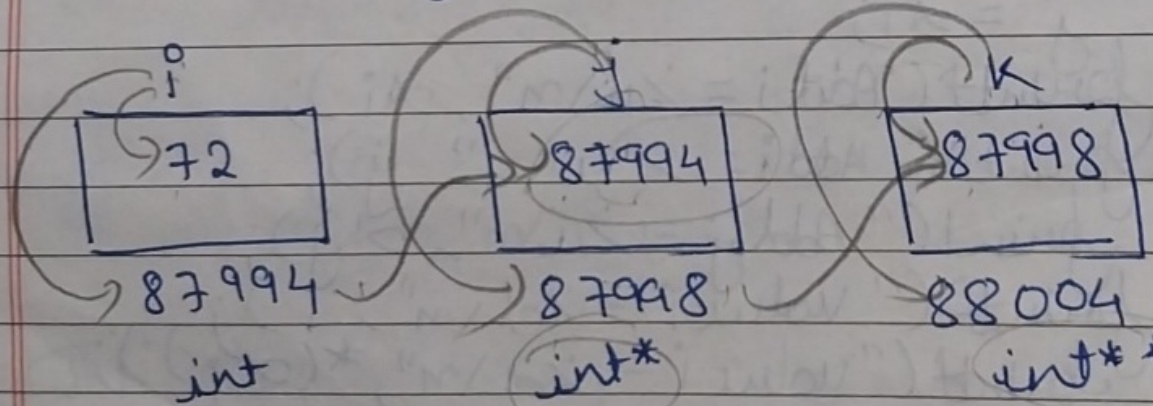
Add i = 87994
Add i = 87994
Add j = 87998
value i = 8
value i = 8
value i = 8

This program sums it all If you understand if you have got the idea of pointers.

Pointer to a pointer

just like j is pointing to i or storing the address of i, we can have another variable k which can further store the address of j what will be the type of k

```
int **k;
k = &j;
```



we can even go further one level and create a variable l of type int*** to store the address of k we mostly use int* and int** sometimes in real world programs

Type of function Calls

Based on the way we pass arguments to the function, function calls are of two types

(i) Call by value → Sending the values of arguments
Call by reference → Sending the address of argume

Call by Value

Here the value of the argumen are passed to the function Consider this example

int c = Sum (3, 4); ⇒ assume x=3 and y=4
x, y

if Sum is defined as Sum (int a, int b), the values 3 and 4 are copied to a and b Now even if we change a and b, nothing happens to the variable x and y This is call by value.

In C we usually make a call by value

Call by reference

Here the addresses of the variables is passed to the function as arguments
Now since the addresses are passed to the function the function can now modify the value of a variable in calling functio

using * and & operator

Example:-

```
void Swap (int *x, int *y)
{
    int temp;
    temp = *x
    *x = *y
    *y = temp;
}
```

This function is capable of swapping the values passed to it. if a=3 and b=4 before a call to Swap(a,b), a=4 and b=3 after calling Swap

```
int main(){
    int a=3
    int b=4 ⇒ a is 3 and b is 4
    Swap(a,b);
    return 0; ⇒ Now a is 4 and
}                b is 3
```

Chapter 6 Practice Set

Q1) Write a program to print the address of variables. Use this address to get the value of the variable

```
#include <stdio.h>
int main(){
    int i=2;
    int *ptr=&i;
    printf("The address of i is %u\n", &i);
    printf("The value of i is %d\n", *ptr);
    return 0;
}
```

Q2) Write a program having a variable 'i'. Print the address of 'i'. Pass this variable to a function and print its address. Are these addresses same? why?

```
#include <stdio.h>
int main(){
    int i = 2
    int *ptr = &i
    printf("The address of i is %u\n", &i);
    returning_5(ptr);
    return 0;
}
int returning_5(int *ptr){
    printf("The value of ptr is %d\n", ptr);
    printf("The value of ptr is %d\n", *ptr);
    return 5;
}
```

Q3) Write a program to change the value of a variable to ten times of its content value.

```
#include <stdio.h>
void change_to_ten_times(int*);
void change_to_ten_times(int*a){
        *a = *a * 10;
}

int main(){
    int x = 45;
    printf("The value of x is %d\n", x);
    change_to_ten_times(&x);
    printf("The value of x is %d\n", x);
    return 0;
}
```

Q4) Write a program using a function which calculates the Sum and average of two numbers. Use pointers and print the values of Sum and average in main().

```
#include <stdio.h>
int *Sum(int a, int b){
    int s = a+b;
    int *ptr = &s;
    printf("The Sum is %d\n", s);
    return ptr;
}
```

```
float *average(int a, int b){
    float avg = (a+b)/2.0;
    float *ptr = &avg;
    printf("The average is %f\n", avg);
    return ptr;
}
int main(){
    int x = 4
    int y = 6;
    int *ptr1;
    float *ptr2;
    ptr1 = sum(x, y);
    ptr2 = average(x, y);

printf("The address of sum is %u and of
        average is %u", ptr1, ptr2);
return 0;
}
```

Q5. Write a program to print the value of variable 'i' by using "pointer to pointer" type of variable.

Ans
```
#include<stdio.h>
int main(){
    int i = 2;
    int* ptr1 = &i;
    int** ptr2 = &ptr1;
    printf("The address of i is %u\n", &i);
    printf("The value of i is %d\n", *ptr1);
    printf("The value of i is %d", **ptr2);
    return 0;
}
```