Address Arithmetic

* Add
* Subtract

Let's take an example

#include<stdio.h>

int main(){

int x;

printf("Address of x is %lu\n", &x);

printf("Address of x) + 1 is %lu\n", &x+1);

printf("Address of x is %lu\n",&x);

printf("Address of x) - 10 is %lu\n", &x-10);

return 0;

}

/\*

Address of x is 6132920648

Address of x) + 1 is 6132920652

Address of x is 6132920648

Address of x) - 10 is 6132920608

\*/

Here when we will write

&x + 1 it will not increase value by 1

That will be changed to &x + (sizeof(x) x 1)

If we write &x - 10 that menas &x - (sizeof(x)x10)

Address Calculation for Array

← – –– –– –– –– –– –– –– –– –– –– –– X –– –– –– –– –– –– –– –– –– –– –– –– –– –– – →

700 (20 Bytes)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |

X[0] x[1] x[2] x[3] x[4]

FAQ

| **What is address of x[0]** | **700** |
| --- | --- |
| **What is data type of x[0]** | **int** |
| What is data type of (&x[0]) address of x[0] | Int \* |
| **What is address of x?** | **700** |
| **What is data type of x?** | **Int [5]** |
| What is data type of address of x ( &x ) | Int [5]\* or int \* [5] |

In which manner you are going to use variable will impect the address

Lets take example

#include<stdio.h>

int main(){

int x[5];

printf("Address of x[0] is %lu\n", &x[0]);

printf("Address of x[0]) + 1 is %lu\n", &x[0]+1);

printf("Address of x is %lu\n", &x);

printf("Address of x) + 1 is %lu\n", &x+1);

return 0;

}

/\*

Address of x[0] is 6137131316

Address of x[0]) + 1 is 6137131320 // Differnce of 4

Address of x is 6137131316

Address of x) + 1 is 6137131336 // difference of 20

\*/

Here we can see when we increase the address of x[1] with 1 then it will increase by 4 because its data type was int

But when we increase the address of x by 1 then it increase the value by 20

Because its data type was int [5]

&x[0] + (sizeof(x[0]) x 1)

&x + (sizeof(x) x 1)

Lets take one more example to see x and x[0] are different

int main(){

int \*p; // p is capable of storing address of a block whose data type is int

int x[5];

p = &x[0]; // correct, p is of type int \* and &x[0] is of type int \*

p = &x; // Incorrect, p is of type int \* and &x is of type int (\*)[5]

return 0;

}

/\*

warning: incompatible pointer types assigning to 'int \*' from 'int (\*)[5]' [-Wincompatible-pointer-types]

p = &x; // Incorrect, p is of type int \* and &x is of type int (\*)[5]

^ ~~

\*/

The benefit of Pointer arithmetic

#include<stdio.h>

int sum(int \*, int);

int main(){

int x[5];

int y,total;

for(y=0; y<=4 ; y++){

printf("Enter a number: ");

scanf("%d", &x[y]);

}

total = sum(&x[0], 5);

printf("Total is %d\n", total);

return 0;

}

int sum(int \*p, int size){

int y,t;

for(y=0,t=0; y<size; y++, p++){

t = t+\*p;

}

return t;

}

How to determine length of array in case of dynamic array length.

#include<stdio.h>

int main(){

int x[5]; // if someone give Dynamic value

// find size of array

int size;

size = sizeof(x)/sizeof(int);

printf("Size is %d\n", size);

return 0;

}