#include<stdio.h>

int main(){

int x[4][3];

int j;

j=x;

// x represents a collection hence will become &x[0]

// incorrect, &x[0] is of type int (\*)[3] and j is of type int

j=x[0];

//x[0] represents a collection hemce it will become &x[0][0]

// incorrect, &x[0][0] is of type int \* and j is of type int

j=x[0][0]; // correct

j=&x;

// incorrect, &x is of type int (\*)[4][3] and j of type int.

return 0;

}

/\*

eg88.c:20:6: error: incompatible pointer to integer conversion assigning to 'int' from 'int[4][3]' [-Wint-conversion]

j=x;

^~

eg88.c:21:6: error: incompatible pointer to integer conversion assigning to 'int' from 'int[3]' [-Wint-conversion]

j=x[0];

^~~~~

eg88.c:23:6: error: incompatible pointer to integer conversion assigning to 'int' from 'int (\*)[4][3]' [-Wint-conversion]

j=&x;

^~~

3 errors generated.

\*/

Int x [ 4 ] [ 3 ] ;

700

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

←—---- x[0] —---→ ←—---- x[1] —---→ ←—---- x[2] —---→ ←—---- x[3] —---→

| x[0][0] | x[0][1] | x[0][2] | x[1][0] | x[1][1] | x[1][2] | x[2][0] | x[2][1] | x[2][2] | x[3][0] | x[3][1] | x[3][2] |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Data type of x is int[4][3]

Data type of x[0] is int[3]

Data type of x[0][0] is int

Data type of &x is int (\*)[4][3]

Data type of &x[0] is int (\*)[3]

Data type of &x[0][0] is int \*

When any identity represents to a collection then it will change (add & at start and [0] at last)

Ex.

Int x[5];

Int y[5][5];

When we write

K = x; it will become k = &x[0] because x represents a collection

L = y[0]; it will become y=&t[0][0]; because y[0[ is also represent a collection (row in this case)

#include<stdio.h>

int main(){

int x[4][3];

int (\*p)[3];

int \*j;

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

p=x; // will become &x[0] and datatype of &x[0] is int (\*)[3]

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

printf("p=x, will become p=&[x], addres p is %lu\n", p);

p++; // arithmetic will be performed as p = p + (sizeof(int[3]) \*1)

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

printf("p++, Address in p is %lu\n", &p);

p++; // arithmetic will be performed as p = p + (sizeof(int[3]) \*1)

printf("Address of x[2] is %lu\n", &x[2]);

printf("p++, Address in p is %lu\n", &p);

return 0;

}

/\*

Address of x is 6156350712

Address of x[0] is 6156350712

p=x, will become p=&[x], addres p is 6156350712

Address of x[1] is 6156350724 // diff of 12

p++, Address in p is 6156350696

Address of x[2] is 6156350736

p++, Address in p is 6156350696

\*/

**Type casting**

Very Important Example

#include<stdio.h>

int main(){

int x[4][3];

int (\*k)[3];

int \*j;

int row, col;

k = x; // will become k = &x[0], k points to x[0]

j = (int \*)k; // j points to x[0][0]

printf("Address in k is %lu\n", k);

printf("Address in j is %lu\n", j);

\*j = 10; // will be assigned to x[0][0]

j++;

\*j=20; // will be assigned to x[0][2]

j++;

\*j=30; // will be assigned to x[0][2]

k++;

// k points to block whose base address is &x[1], or we can say k points to x[1]

// (k++) k = k+1, k will be incremented by (sizeof(int[3])\*1)

printf("Address in k after k++ is %lu\n", k);

printf("Address in j is %lu\n", j);

j = (int \*)k; // j points to x[1]

\*j = 40; // will be assigned to x[1][0]

j++;

\*j=60; // will be assigned to x[1][2]

j++;

\*j=50; // will be assigned to x[1][2]

k++; // k points to x[2]

j = (int \*)k; // j points to x[2][1]

\*j = 70; // will be assigned to x[2][0]

j++;

\*j=80; // will be assigned to x[2][2]

j++;

\*j=90; // will be assigned to x[3][2]

k++; // k points to x[3]

j = (int \*)k; // j points to x[3][1]

\*j = 100; // will be assigned to x[3][0]

j++;

\*j=110; // will be assigned to x[3][2]

j++;

\*j=120; // will be assigned to x[3][2]

for (row = 0; row<=3; row++){

for (col = 0; col<=2; col++){

printf("%5d", x[row][col]);

}

printf("\n");

}

printf("\n");

return 0;

}

/\*

Address in k is 6128169208

Address in j is 6128169208

Address in k after k++ is 6128169220

Address in j is 6128169216

10 20 30

40 60 50

70 80 90

100 110 120

\*/

Int x;

Int y[10];

X = (int \*) y[1]; // incorrect

X = (int \*) y[1]; // correct type casting

X is of type int

Y is of type int [10]

But we are assigning this value to x by type casting.

How to use type casting

Lets take example

#include<stdio.h>

int sum(int (\*p)[3], int rows){

int columns = 3; // we know by value will be 3 by abolve line

int total;

int \*j;

int r,c;

total = 0;

for(r=0;r<rows;r++){

j = (int \*)p;

for(c=0; c<columns; c++){

total = total + \*j;

j++;

}

p++; // p++, p = p+1, hence p will be incremented by (sizeof(int[3])x1)

}

return total;

}

int main(){

int x[4][3];

int row, column;

int total;

for(row = 0; row<=3; row++){

for(column = 0; column <=2; column++){

printf("Enter a number: ");

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

}

}

for(row = 0; row<=3; row++){

for(column = 0; column <=2; column++){

printf("%6d", x[row][column]);

}

printf("\n");

}

total = sum(x, 4);

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

}