This Feature makes things convenient means a more readable code format.

But technically array will not going to pass

Everything will be converted into pointers internally.

Int sam(int \*P)

We can write it as

**int sam (int p[])** but this will be converted into Int sam(int \*P)

We know at the time of cal sam (x) treated as sam (&x[0]) which have type int \*;

And if 2d array then type will be int (\*)[3] ; //

Lets see example

#include<stdio.h>

void sam(int p[]){ // should be treated as int \*p

printf("%lu\n", sizeof(p));

// some code

}

int main(){

int x[5];

// some code

sam(x);

// some code

return 0;

}

/\*

Compile response:

warning: sizeof on array function parameter will return size of 'int \*' instead of 'int[]' [-Wsizeof-array-argument]

printf("%lu\n", sizeof(p));

^

eg117.c:2:14: note: declared here

void sam(int p[]){

^

1 warning generated.

Output:

8

\*/

It shoule be like

#include<stdio.h>

void sam(int p[], int sz){ // should be treated as (int \*p, int sz)

printf("%lu\n", sizeof(p));

// some code

}

int main(){

int x[5];

// some code

sam(x, 5); // ideally we should pass the size otherwise how can we determine the size?

// some code

return 0;

}

/\*

Compile response:

warning: sizeof on array function parameter will return size of 'int \*' instead of 'int[]' [-Wsizeof-array-argument]

printf("%lu\n", sizeof(p));

^

eg117.c:2:14: note: declared here

void sam(int p[]){

^

1 warning generated.

Output:

8

\*/

To get the length of the pointer.

Example of 2d

#include<stdio.h>

void sam(int p[][3]){ // shoule be trated as sam(int (\*p)[3])

printf("%lu\n", sizeof(p));

}

int main(){

int x[5][3];

// some code

sam(x); // should be treated as sam(&x[0])

// some code

return 0;

}

/\*

Compiler:

eg118.c:3:27: warning: sizeof on array function parameter will return size of 'int (\*)[3]' instead of 'int[][3]' [-Wsizeof-array-argument]

printf("%lu\n", sizeof(p));

^

eg118.c:2:14: note: declared here

void sam(int p[][3]){ // shoule be trated as sam(int (\*p)[3])

^

1 warning generated.

output:

8

\*/

It should be like to get rows

#include<stdio.h>

void sam(int p[][3], int rows){ // shoule be trated as sam(int (\*p)[3], int rows)

printf("%lu\n", sizeof(p));

}

int main(){

int x[5][3];

// some code

sam(x, 5); // should be treated as sam(&x[0]) // we should pass rows to get the size of rows

// some code

return 0;

}

/\*

Compiler:

eg118.c:3:27: warning: sizeof on array function parameter will return size of 'int (\*)[3]' instead of 'int[][3]' [-Wsizeof-array-argument]

printf("%lu\n", sizeof(p));

^

eg118.c:2:14: note: declared here

void sam(int p[][3]){ // shoule be trated as sam(int (\*p)[3])

^

1 warning generated.

output:

8

\*/

If we pass the sam(int p[5]) length as a parameter then its insignificant it will be removed

It will be treated as int \*p only

Example

#include<stdio.h>

//The following example is useless unless int sz is added to the list of parameters.

void sam(int p[5]){ // should be treated as void (int \*p)

printf("%lu\n", sizeof(p));

// some code

}

int main(){

int x[5];

// some code

sam(x);

// some code

return 0;

}

/\*

Compile response:

warning: sizeof on array function parameter will return size of 'int \*' instead of 'int[5]' [-Wsizeof-array-argument]

printf("%lu\n", sizeof(p));

^

eg119.c:2:14: note: declared here

void sam(int p[5]){ // should be treated as int \*p

^

------> Here 5 is insignificant, it will be removed.

// eventually same as 117 example.

1 warning was generated.

Output:

8

\*/