Date
Page

C_101 ⇒ Function Pointers in C.

Declaration of function Pointer

```
return-type of function (*pointer-name)(Datatype of
                                        arguments);
```

Normal function declaration & definition

```
int sum(int, int);
```

```
int sum(int a, int b)
{
    return a+b;
}
```

Note :-

* Now we want a pointer which points to this function, it depends on the ~~returntype~~ prototype of the function.

Declaration of function pointer

```
int (*ptr)(int, int);
```

This is declaration of function pointer for above function sum.

* Function pointer will contain address of function

Initialization of function pointer

```
int (*ptr)(int, int) = &sum;
```

Why to put bracket ?

✓ int (*ptr)(int, int);

✗ int *ptr(int, int);

Explanation :-

int *ptr(int, int); <=> int* ptr(int, int);

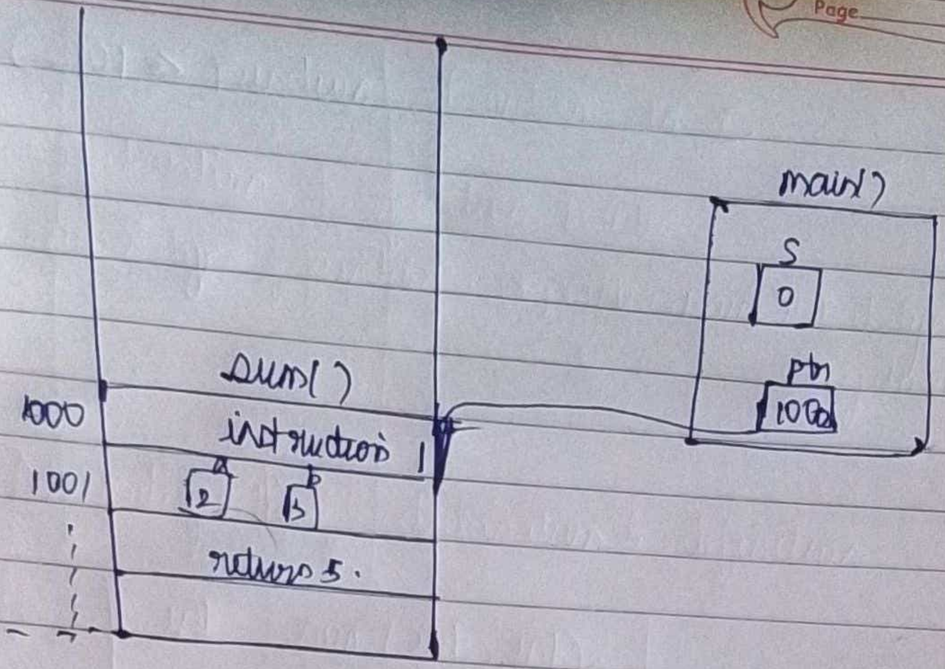
$\underbrace{*}_{\text{low priority}}$ $\underbrace{\text{ptr(int, int)}}_{\text{higher priority}}$

* Compiler takes this ptr ~~is~~ as function and ~~[illegible]~~ accepts two ~~argumen~~ arguments with return type as pointer

* In main call the function using its pointer name instead of function name.

Program :-

```
int sum(int, int);
void main()
{
  int s=0;
  int (*ptr)(int, int) = &sum;
    s=(*ptr)(2,3);
  printf("%d",s);
}
int sum(int a, int b)
{
   return a+b;
}
```

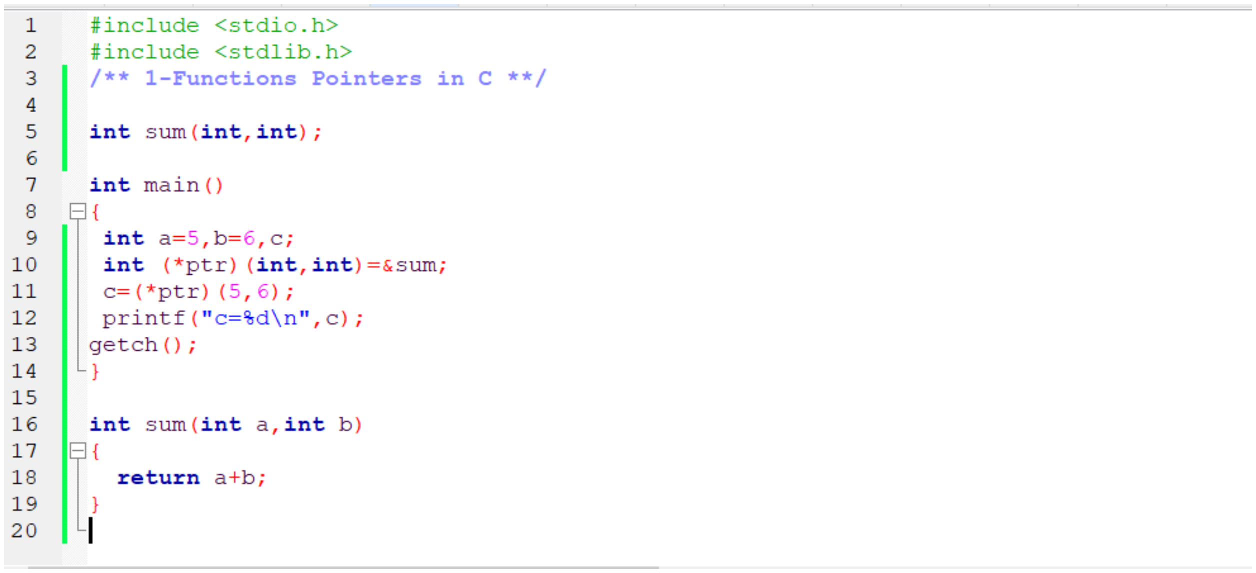


* Function pointer contains address of that code

* So we dereference it and do the work.

```
//int (*ptr)(int, int) = sum;
//      S = ptr(2,3);
```

This is also correct.

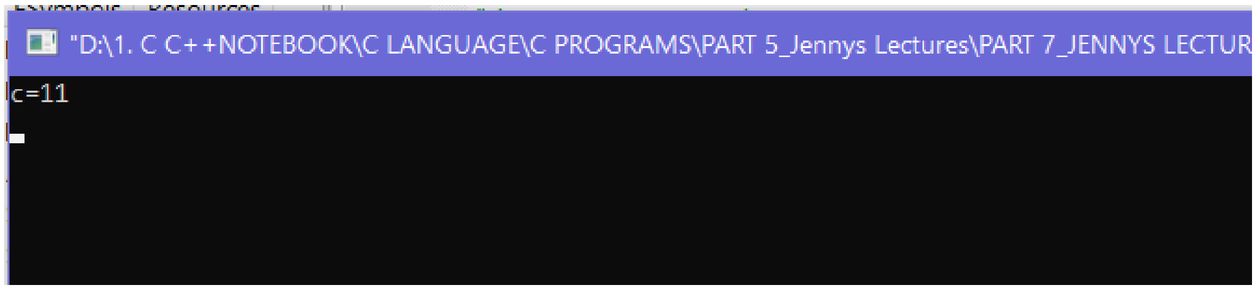


```c
#include <stdio.h>
#include <stdlib.h>
/** 1-Functions Pointers in C **/

int sum(int,int);

int main()
{
 int a=5,b=6,c;
 int (*ptr)(int,int)=&sum;
 c=(*ptr)(5,6);
 printf("c=%d\n",c);
getch();
}

int sum(int a,int b)
{
  return a+b;
}
```



```
"D:\1. C C++NOTEBOOK\C LANGUAGE\C PROGRAMS\PART 5_Jennys Lectures\PART 7_JENNYS LECTUR
c=11
```