Function Pointers

What is a function pointer

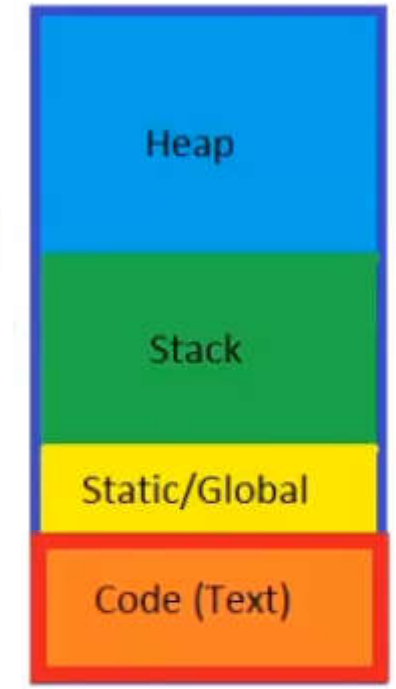
Pointers → Can point to data

→ Can point to functions

Use cases?

What is a function pointer

Application's memory



<< Program.c >>

```
int main()
{
  printf("Hello");
}
```

Source code

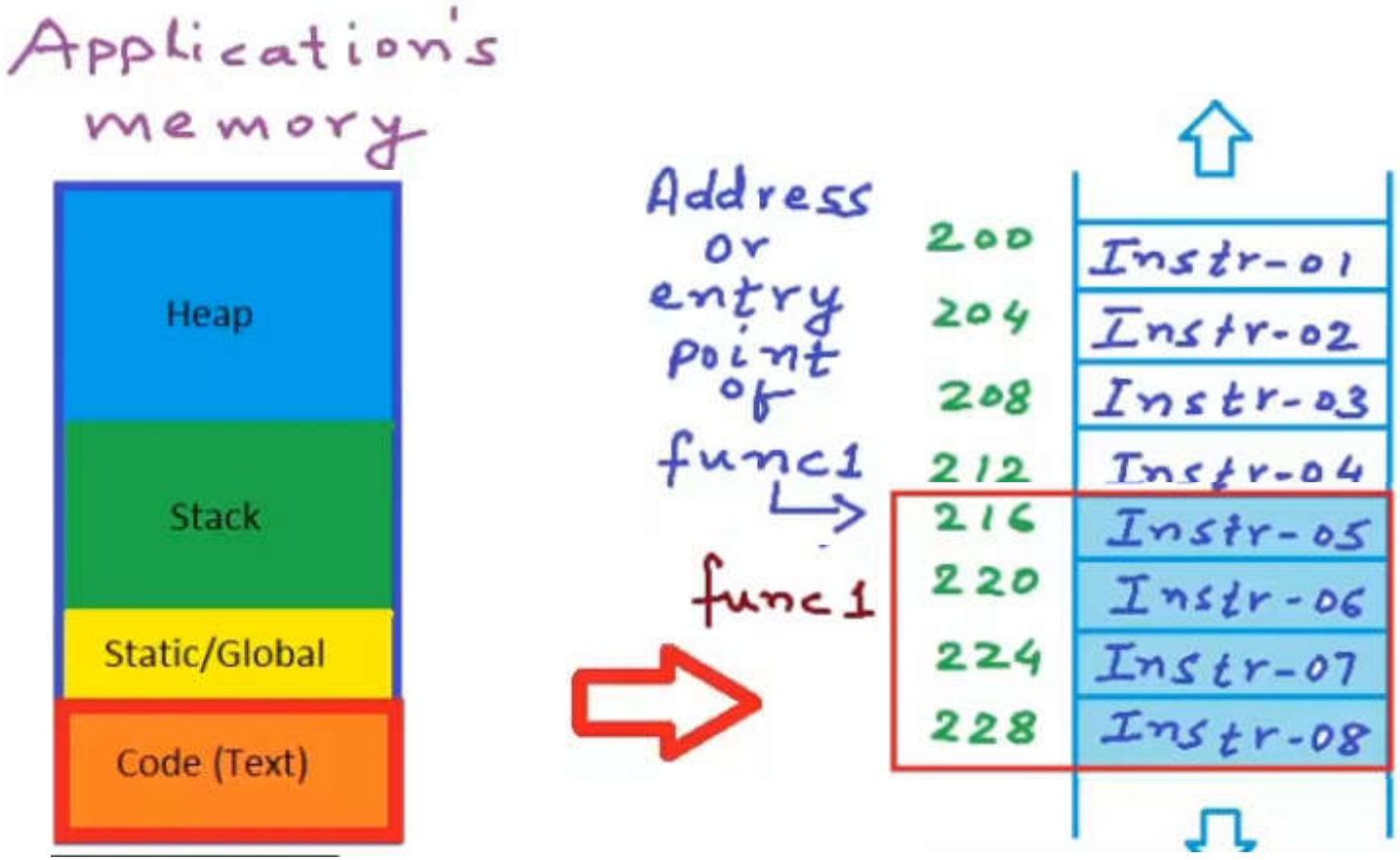
⇒ Compiler ⇒

<< Program.exe >>

```
10010010
11001100
11100011
10000011
10101010
```

Machine code ⇒

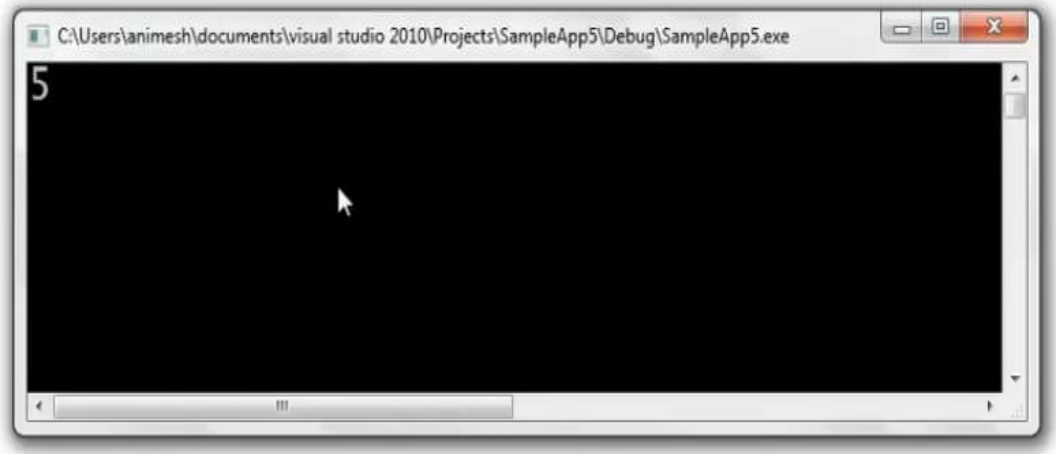
What is a function pointer



Function pointer example

```
#include<stdio.h>
int Add(int a,int b)
{
    return a+b;
}

int main()
{
    int c;
    int (*p)(int,int);
    p = &Add;
    c = (*p)(2,3); //de-referencing
    printf("%d",c);
}
```



Function pointer example

```
#include<stdio.h>
int Add(int a,int b)
{
    return a+b;
}
int *Func(int,int); //declaring a function
int main()
{
    int c;
    int *p(int,int);
    p = &Add;
    c = (*p)(2,3); //de-referencing
    printf("%d",c);
}
```

Function pointer alternative way

```
#include<stdio.h>
int Add(int a,int b)
{
    return a+b;
}
int main()
{
    int c;
    int (*p)(int,int);
    p = &Add; / function name will return us pointer
    c = (*p)(2,3); //de-referencing and executing the function.
    printf("%d",c);
}
```

Function pointer alternative way

```c
#include<stdio.h>
int Add(int a,int b)
{
    return a+b;
}
int main()
{
    int c;
    int (*p)(int,int);
    p = Add; // function name will return us pointer
    c = (*p)(2,3); //de-referencing executing the function.
    printf("%d",c);
}
```

Compilation Error

```
#include<stdio.h>
int Add(int a,int b)
{
    return a+b;
}
int main()
{
    int c;
    void (*p)(int,int);
    p = Add; // function name will return us pointer
    c = p(2,3); //de-referencing and executing the function.
    printf("%d",c);
}
```

Compilation Error

```c
#include<stdio.h>
int Add(int a,int b)
{
    return a+b;
}
int main()
{
    int c;
    int (*p)(int);
    p = Add; // function name will return us pointer
    c = p(2,3); //de-referencing and executing the function.
    printf("%d",c);
}
```

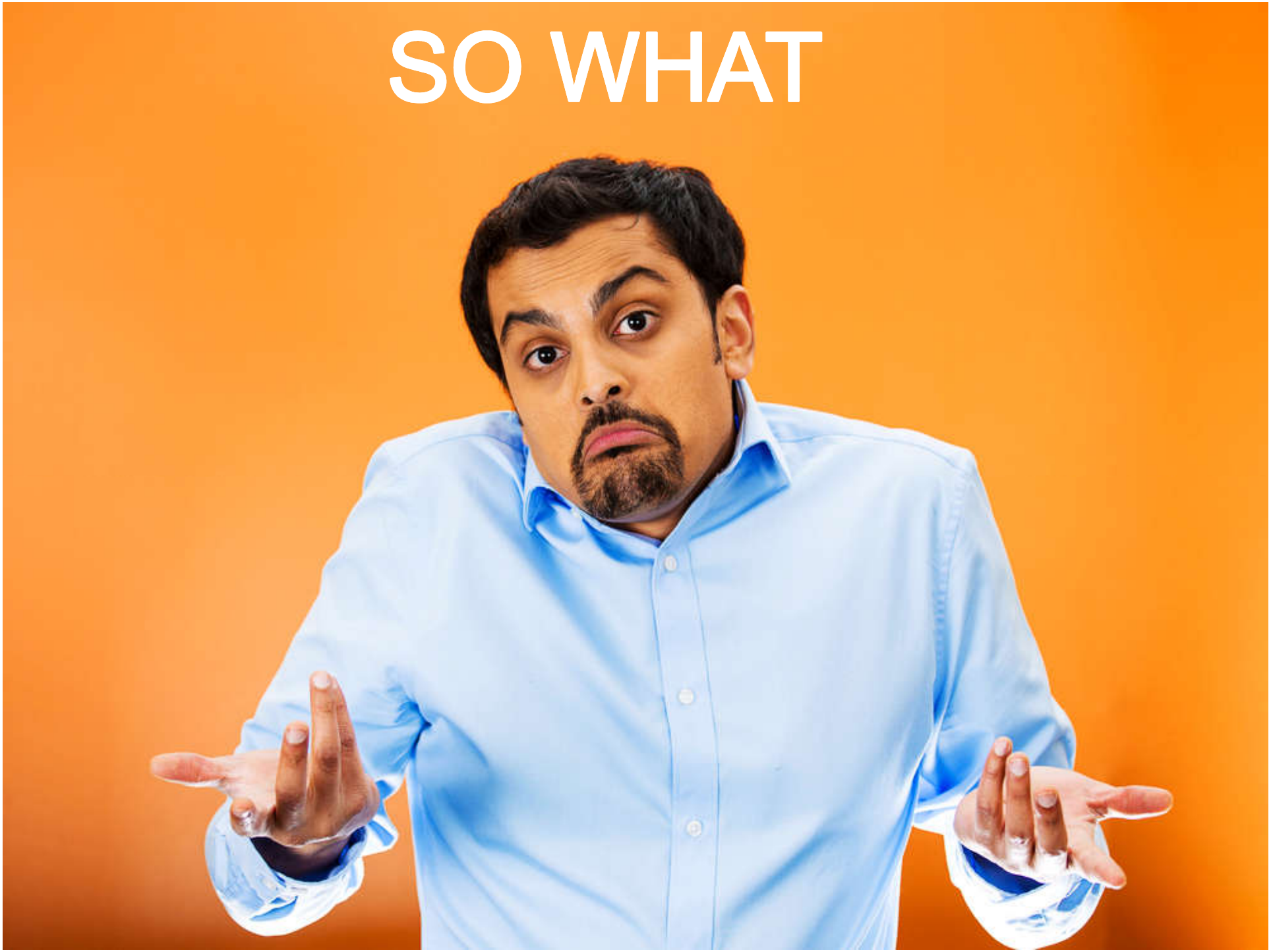
Function pointer another example

```
//Function Pointers in C/C++
#include<stdio.h>
void PrintHello(char *name)
{
    printf("Hello %s\n",name);
}
int Add(int a,int b)
{
    return a+b;
}
int main()
{



}
```

SO WHAT



Function pointer use cases

Function pointers can be passed as an argument to a function

Function receiving the address of a function as parameter can call back the passed function

Function pointer and call back

```c
//Function Pointers and callbacks
#include<stdio.h>
void A()
{
    printf("Hello");
}
int main()
{

}
```

Function pointer and call back

```
//Function Pointers and callbacks
#include<stdio.h>
void A()
{
    printf("Hello");
}
void B(void (*ptr)()) // function pointer as argument
{
   ptr(); //call-back function that "ptr" points to
}
int main()
{  void (*p)() = A;
   B(p);
}
```

B(A);

Call back Sorting example

```
// sort in increasing order => {1,2,3,4,5,6}
```

```
#include<stdio.h>
int main()
{
    int A[] ={3,2,1,5,6,4};
}
```

Call back Sorting example

```
void BubbleSort(int *A,int n) {
    int i,j,temp;
    for(i =0; i<n; i++)
        for(j=0; j<n-1; j++) {
            if(A[j] < A[j+1]) { //compare
                temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
        }
}
int main() {
    int i, A[] ={3,2,1,5,6,4};
    BubbleSort(A,6);
    for(i =0;i<6;i++) printf("%d ",A[i]);
}
```

Call back Sorting example

```
void BubbleSort(int *A,int n,int flag) {
    int i,j,temp;
    for(i =0; i<n; i++)
        for(j=0; j<n-1; j++) {
            if(A[j] < A[j+1]) { //compare
                temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
        }
}
int main() {
    int i, A[] ={3,2,1,5,6,4};
    BubbleSort(A,6);
    for(i =0;i<6;i++) printf("%d ",A[i]);
}
```

Call back Sorting example

```
void BubbleSort(int *A,int n,int (*compare)(int,int)) {
    int i,j,temp;
    for(i =0; i<n; i++)
        for(j=0; j<n-1; j++) {
            if(compare(A[j],A[j+1])> 0) { // compare
                temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
        }
}
int main() {
    int i, A[] ={3,2,1,5,6,4};
    BubbleSort(A,6);
    for(i =0;i<6;i++) printf("%d ",A[i]);
}
```

```c
#include<stdio.h>
int compare(int a,int b)
{
    if(a > b) return 1;
    else return -1;
}
void BubbleSort(int *A,int n,int (*compare)(int,int)) {
    int i,j,temp;
    for(i =0; i<n; i++)
        for(j=0; j<n-1; j++) {
            if(compare(A[j],A[j+1]) > 0) { //compare A[j]
                temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
        }
}
int main() {
    int i, A[] ={3,2,1,5,6,4};
    BubbleSort(A,6,compare);
    for(i =0;i<6;i++) printf("%d ",A[i]);
}
```

```c
int absolute_compare(int a,int b)
{
    if(abs(a) > abs(b)) return 1;
    return -1;
}
void BubbleSort(int *A,int n,int (*compare)(int,int)) {
    int i,j,temp;
    for(i =0; i<n; i++)
        for(j=0; j<n-1; j++) {
            if(compare(A[j],A[j+1]) > 0) { //compare A[j] with
                temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
        }
}
int main() {
    int i, A[] ={-31,22,-1,50,-6,4}; // => {-1,4,-6,22,-31,50}
    BubbleSort(A,6,absolute_compare);
    for(i =0;i<6;i++) printf("%d ",A[i]);
}
```

qsort

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int main() {
    int i, A[] ={-31,22,-1,50,-6,4};
    qsort(A,6,sizeof(int),
}
```

qsort

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int compare(const void* a, const void* b)
{


}
int main() {
    int i, A[] ={-31,22,-1,50,-6,4}; // =>
    qsort(A,6,sizeof(int),
}
```

qsort complete picture

```c
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int compare(const void* a, const void* b)
{
    int A = *((int*)a); // tyepcasting to int* and getting value
    int B = *((int*)b);
    return A-B;
}
int main() {
    int i, A[] ={-31,22,-1,50,-6,4}; //
    qsort(A,6,sizeof(int),compare);
    for(i = 0;i<6;i++) printf("%d ",A[i]);
}
```

Now you know Function Pointers


The End