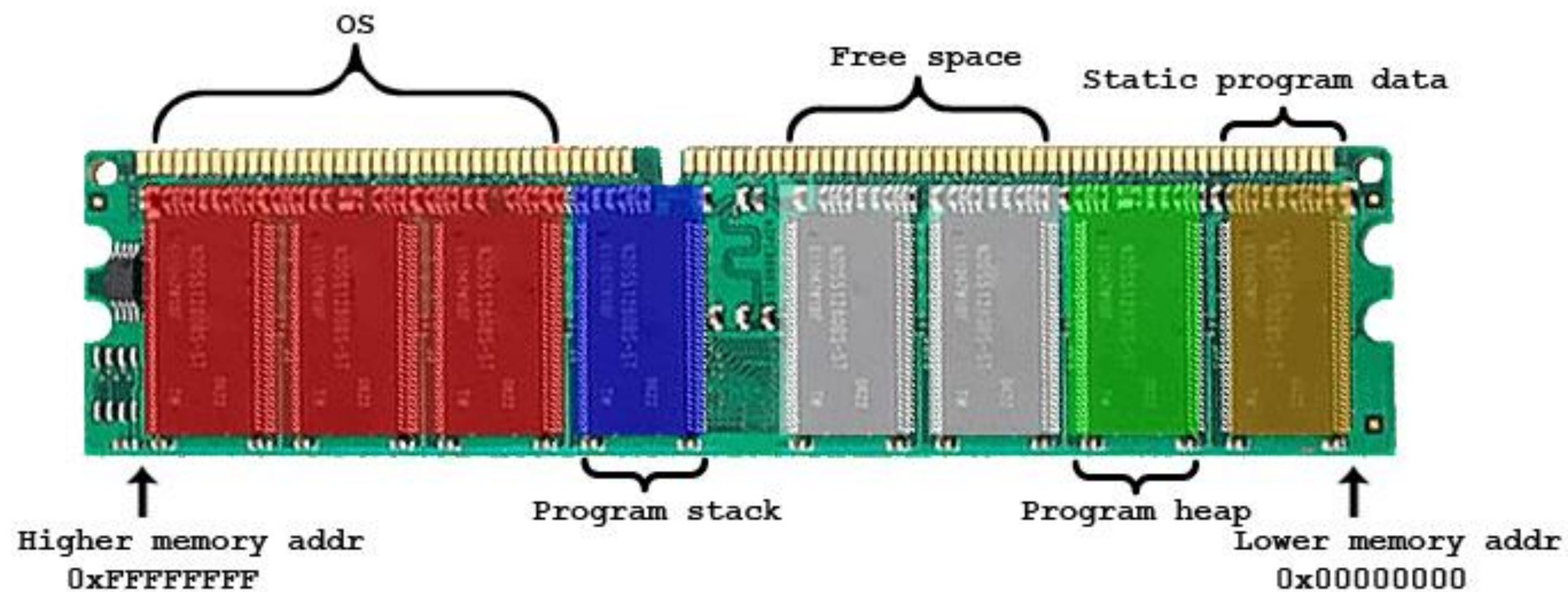


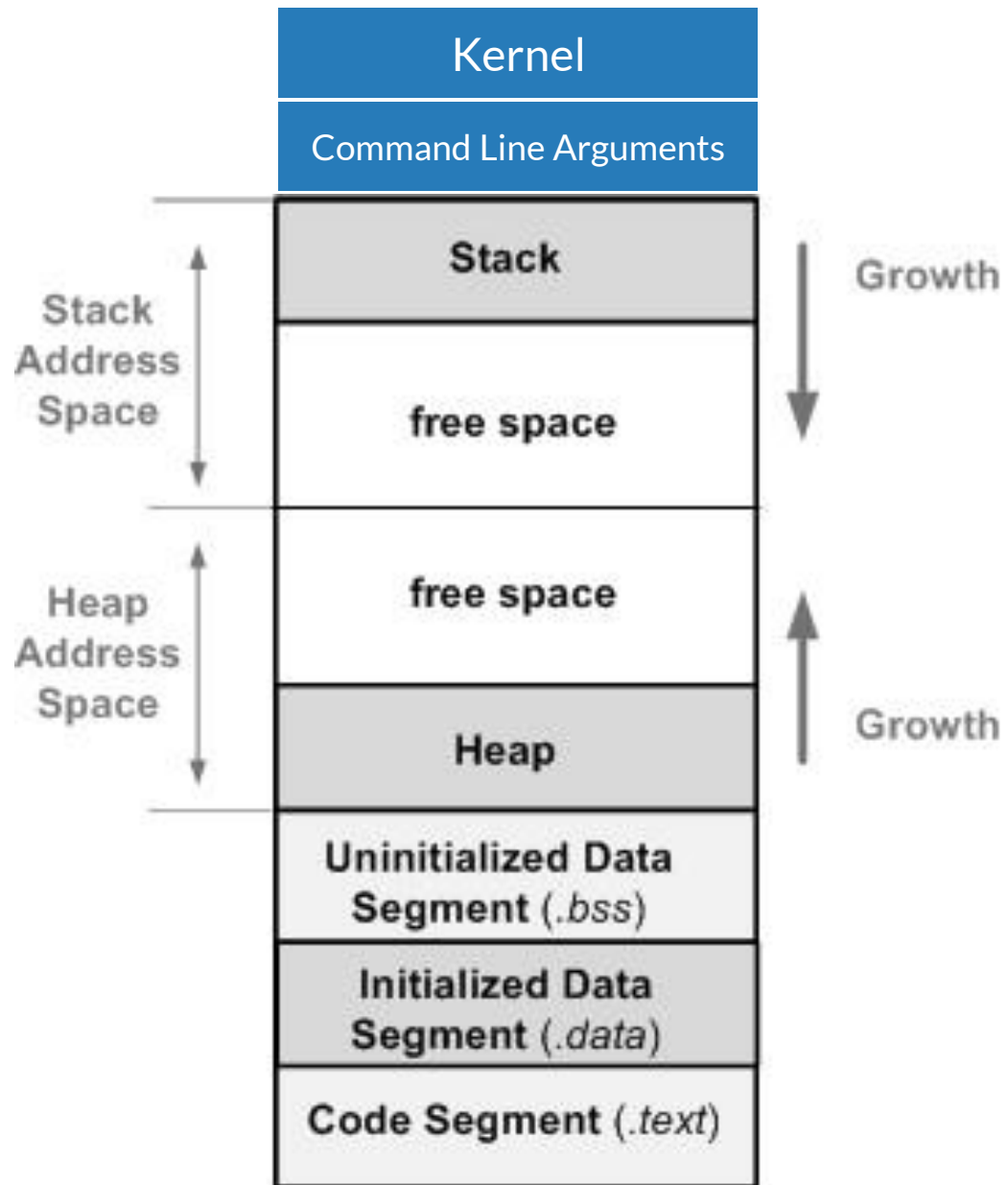


Programing Workshop For Beginners

جلسه نهم

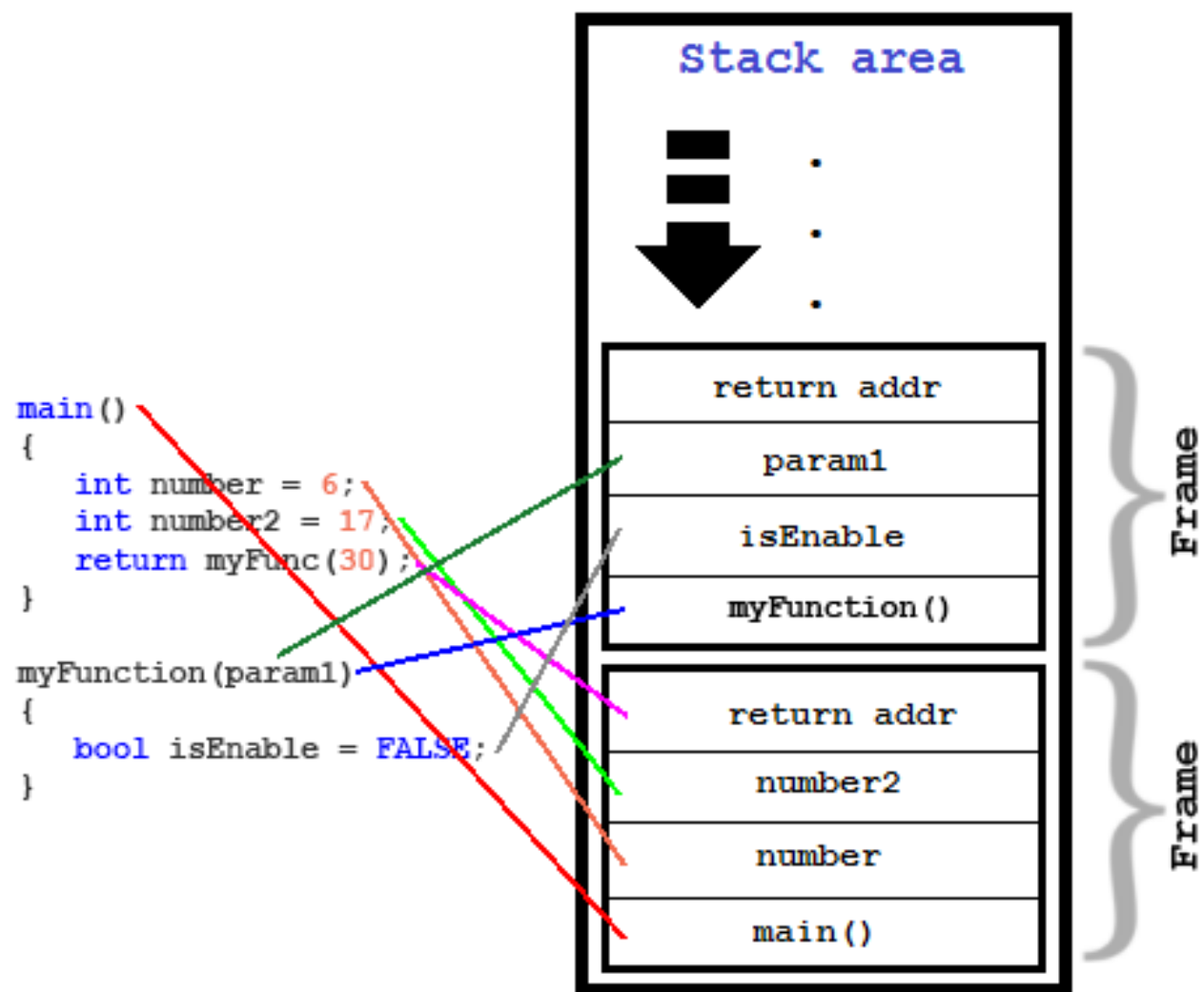
1399 - 1400



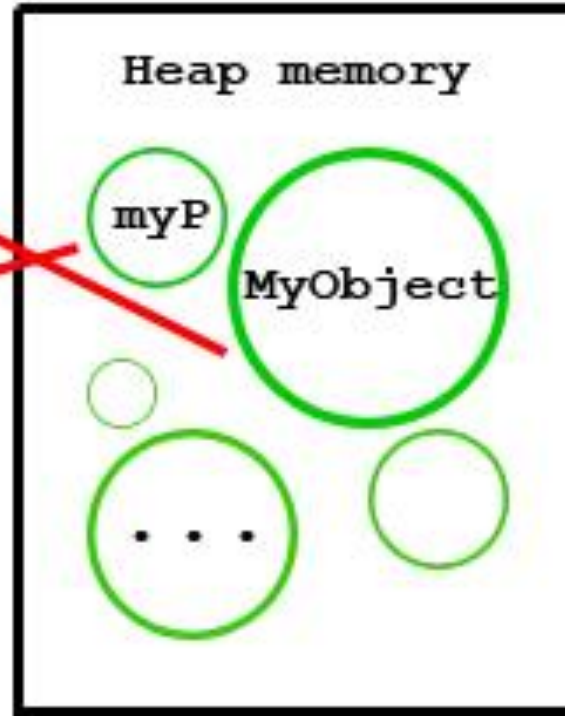


وقتی متغیر محلی **local variable** یا تابعی تعریف و استفاده می کنید مقادیر و آدرسشون در قسمت **Stack** حافظه قرار می گیره

ولی با ساخت شی یا اختصاص حافظه بصورت دستی **Dynamic**، مقدار و آدرسشون در قسمت **Heap** حافظه مجازی قرار میگیره.





```
int myfunction()  
{  
    int i;  
  
    MyObject object = new MyObject();  
  
    int *myP = (int*)malloc(10);  
  
    return i  
}
```



malloc()
calloc()
realloc()
free()

```
//  
// Created by jaberALU on 03/01/2021.  
//  
  
#include <stdio.h>  
  
char c;          /* Uninitialized variable */  
int main() {  
    static int i; /* Uninitialized static variable */  
    return 0;  
}
```

```
// Created by jaberALU on 03/01/2021.  2  3  
//  
  
#include <stdio.h>  
  
char c[] = "jaber babaki"; /*global variable stored in Initialized Data Segment  
int main() {  
    static int i = 11; /* static variable stored in Initialized Data Segment*/  
    return 0;  
}
```

اشاره‌گر به نوعی از داده می‌گویند که به محل ذخیره داده‌ای دیگر بر روی حافظه اشاره می‌کند و به محتویات آن داده دسترسی دارد.

```
type *name;
```

```
int *ptr;
```

```
int * ptr;
```

```
int* ptr;
```

```
int* p,q;
```

```
int a = 5;
```

```
int *ptr;
```

```
ptr = &a;
```

اشاره‌گر به نوعی از داده می‌گویند که به محل ذخیره داده‌ای دیگر بر روی حافظه اشاره می‌کند و به محتویات آن داده دسترسی دارد.

```
int *mp;  
int b;  
mp = &b;  
*mp = 3;
```

اشاره‌گر در زبان C یک آدرس می‌باشد که این آدرس به صورت عددی است. بنابراین می‌توان عملیات ریاضی را بر روی آن انجام داد.

```
int var=10;  
int *ptr;  
ptr = &var;  
for ( i = 3; i > 0; i--) {  
    printf("Address\n ", ptr );  
    printf("Value\n ", *ptr );  
    ptr--;  
}
```



```
int var = 10;
int b = 10;
int *ptr;
ptr = &var;
if (ptr == &b) {
    printf("equal");
}
```

آرایه‌ها به واسطه اشاره‌گرها در کامپایلر نوشته می‌شوند

```
int a[5];
a[0]=1,a[1]=10,a[2]=10;
printf("%d\n",*(a+2));
printf("%d\n",(a+2));
printf("%d\n",&a[2]);
```

```
Int a[5];  
int var;  
a[0]=&var;
```

```
int arr[5] = { 1, 2, 3, 4, 5 };  
int *ptr = arr;  
printf("%d\n", (*ptr+1));
```

```
int a[2][2];  
a[0][0]=86;a[0][1]=2;a[1][1]=3;  
printf("%d\n",*(*(a+1)+1));
```

`int *ptr[10];`

`int = integer`

`int * = pointer-to-integer`

`int ** = pointer-to-(pointer-to-integer)`

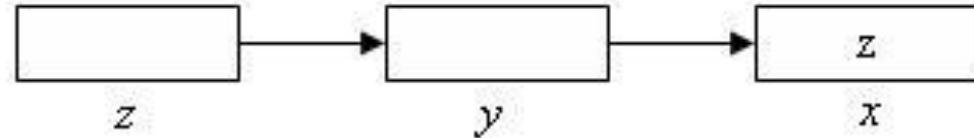
`int *** = pointer-to-(pointer-to-(pointer-to-integer))`

`int ***ptr;`

`int i = 812;`

`int *ptr = &i;`

`int **ptr2 = &ptr;`



```
int k = 5, m = 8;  
int *ptr = &k;  
int **ptr2 = &ptr;
```

```
**ptr2 = 12;  
*ptr2 = &m;
```

```
int k = 5, m = 8;  
int *ptr = &k;  
int **ptr2 = &ptr;  
**ptr2 = 12;  
*ptr2 = &m;
```

```
char *Msg1 = "This is a message";  
printf("%c",*(Msg1+1));
```

فرآیند ی برای تخصیص حافظه وجود دارد که به شما اجازه می دهد که تصمیم گیری برای اندازه آرایه را به زمان اجرای برنامه runtime موکول کنید. این فرآیند “تخصیص حافظه پویا” نام دارد.

تابع	هدف
<u>malloc</u>	حافظه با اندازه مطلوب را اختصاص داده و یک اشاره گر به اولین بیت فضای تخصیص یافته را برمی گرداند.
<u>calloc</u>	به عناصر یک آرایه فضا تخصیص می دهد. عناصر را با صفر مقداردهی اولیه کرده و یک اشاره گر به حافظه برمی گرداند.
<u>realloc</u>	برای ویرایش اندازه حافظه ای که قبلاً تخصیص داده شده بکار می رود.
Free	حافظه ای که قبلاً تخصیص یافته را خالی و یا آزاد می سازد.

فرآیندی برای تخصیص حافظه وجود دارد که به شما اجازه می دهد که تصمیم گیری برای اندازه آرایه را به زمان اجرای برنامه runtime موکول کنید. این فرآیند “تخصیص حافظه پویا” نام دارد.

```
int *ptr;
ptr = malloc(15 * sizeof(*ptr)); /* a block of 15 integers */
if (ptr != NULL) {
    *(ptr + 5) = 480; /* assign 480 to sixth integer */
    printf("Value of the 6th integer is %d",*(ptr + 5));
}
```

```
int i, *ptr, sum = 0;
ptr = calloc(10, sizeof(int));
for (i = 0; i < 10; ++i) {
    *(ptr + i) = i;
    sum += *(ptr + i);
}
printf("Sum = %d", sum);
```



THANK YOU!
FOR YOUR ATTENTION