

# Day3\_Cprogramming

## Day3

### Type Hirericy

#### Constrol Statement

- Loop

- Open Loop

- Close Loop

Arrays

String

1

في حالة استخدام أي Operation ينتج عنها Expression وهو قيمة يجب ان يكون لها نوع بيانات

## Size Point

## Type Hieriricy



char --> 1byte --> 255

int --> 4byte --> 2b

float --> 4byte 2b.7p

double --> 8Byte 48

Float + int = Float

int + int = int

int + char = int

int + double = double

double + char = double

double + float = double

2

## - Loop

## - Close Loop

في حالة وجود معلومية عدد مرات تكرار اللوجيك

## For Loop

for( Start Point;Condtion; Increment - Decement )

```
{
    //Logic
}
```

```
float Balance= 1000;
int Years = 5;
float Interest = 0.1;
float Amount;

Amount = Balance * Interest;//100
Balance += Amount;//1100
printf("Amount In First Year = %f, Balanvce = %f", Amount , Balance)
```

```
Amount = Balance * Interest;//110
Balance += Amount;//1210
printf("Amount In Secound Year = %f, Balanvce = %f", Amount , Balance)
```

```
Amount = Balance * Interest;
Balance += Amount;
printf("Amount In Third Year = %f, Balanvce = %f", Amount , Balance)
```

```
Amount = Balance * Interest;
Balance += Amount;
printf("Amount In Forth Year = %f, Balanvce = %f", Amount , Balance)
```

```
Amount = Balance * Interest;
Balance += Amount;
printf("Amount In Fifth Year = %f, Balanvce = %f", Amount , Balance)
```

1. Years = n
2. اللوجيك داخل Body ال For هيتنفذ n من المرات
3. increment or decrement هيتنفذ n من المرات
4. فحص الشرط هيتنفذ n+1 من المرات
5. Initlization - Start Point هتتنفذ مرة واحدة فقط

3

```
1 for(int i = 0; i < Years; i++)
2 {
3     Amount = Balance * Interest;
4     Balance += Amount;
5     printf("Amount Of Ye %i = %.1f, Balance = %.1f\n", i+1, Amount, Balance);
6 }
```

## Swap Variable That Equal 123 to 321;

```
int x = 123;
```

```
int y=0;
```

```
y = y + x % 10;
```

$$x = x / 10$$

```
y = y * 10;
```

```
y = y + x % 10;
```

$$x = x / 10$$

```
y = y * 10;
```

```
y = y + x % 10;
```

$x = x / 10 // 12$

```
y = y * 10;
```

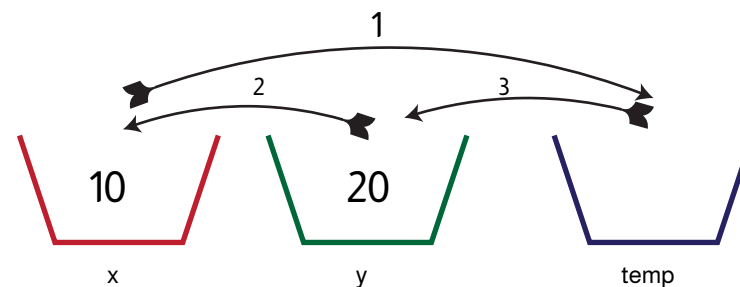
V1 - Very Bad

```
1  int x = 123;
2  int y=0;
3
4  y = y + x % 10;
5  x = x / 10
6  y = y * 10;
7
8  y = y + x % 10;
9  x = x / 10
10 y = y * 10;
11
12 y = y + x % 10;
13 x = x / 10//12
14 y = y * 10;
```

**V2 - Good** ↓

```
1 int y=0;
2 for(int i=0;i<9;i++){
3     y = y * 10;
4     y = y + x % 10;
5     x = x / 10;
6     printf("x = %i\n",x);
7     printf("y = %i\n",y);
8     printf("\n\n_____\n\n");
9 }
```

4



```
1 int x = 10;
2 int y = 20;
3
4 printf("x = %i\n",x);
5 printf("y = %i\n",y);
6 //V1
7 temp = x;
8 x = y;
9 y = temp;
10 //V2
11 x = x + y;//30;
12 y = x - y;//10
13 x = x - y;//20
14 //V3
15 x = x * y;//30;
16 y = x / y;//10
17 x = x / y;//20
18 printf("x = %i\n",x);
19 printf("y = %i\n",y);
```

5

## Nested If

```

1
2 int age =16;
3 char gender ='m';
4
5 if(age>=16){
6
7     if(gender =='m'){
8         //Allow Male
9     }else{
10        //Allow Female
11    }
12 }

```

## Nested For

```

1 for(int i = 0; i<5; i++){
2
3     for(int j = 0; j<5; j++){
4         //logic
5     }
6 }
7

```

```

1 for(int i = 0; i<5; i++){
2
3     for(int j = 0; j<5; j++){
4         printf("*");
5     }
6     printf("\n");
7
8 }

```

```

1 for(int i = 0; i<5; i++){
2
3     for(int j = 0; j<i+1; j++){
4         printf("*");
5     }
6     printf("\n");
7 }
8

```

Loop Num1	Loop Num2	Loop Num3	Loop Num3	Loop Num3
i = 0	i = 1	i = 2	i = 3	i = 4
j = 0	j = 0	j = 0	j = 0	j = 0
j = 1	j = 1	j = 1	j = 1	j = 1
j = 2	j = 2	j = 2	j = 2	j = 2
j = 3	j = 3	j = 3	j = 3	j = 3
j = 4	j = 4	j = 4	j = 4	j = 4
j = 5	j = 5	j = 5	j = 5	j = 5

```

      j j j j j
i *****
i *****
i *****
i *****
i *****

```

```

*****
*****
***
**
*

```

7

```

      j
i      *
i      **
i      ***
i      ****
i      *****

```

```

1 for(int i = 0; i<5; i++){
2
3     for(int j = 5; j>i; j--){
4         printf("*");
5     }
6     printf("\n");
7 }

```

- Loop

- Open Loop

فى حالة عدم وجود معلومية عدد مرات تكرار اللوجيك

While

While(Condition){

//Logic

do While

}

```
1 //Bussiness Logic age > 18
2 printf("Please Enter Your Age\n");
3 scanf("%i",&Age);
4
5 while(Age<=18){
6     printf("Please Enter Age > 18\n");
7     printf("Please Enter Your Age\n");
8     scanf("%i",&Age);
9 }
10 printf("Your Age Is %i\n",Age);
```

# While

القاعدة بتقول ان طول ما الشرط داخل اللوب true = هيدخل ينفذ الBody

2- دائما فى الشرط اكتب عكس البيزنس المطلوب

3- اولاً يفحص الشرط وثانياً ينفذ الكود بداخل الBody

8

# do While

do{

}while();

القاعدة بتقول ان طول ما الشرط داخل اللوب true = هيدخل ينفذ الBody

2- دائما فى الشرط اكتب عكس البيزنس المطلوب

3- اولاً ينفذ الكود داخل الBody وثانياً يفحص الشرط

4- يبقى فى حالة استخدام Do While الكود هيتنفذ مرة واحدة على الاقل حتى اذا كان الشرط = False

```
1 int Age;
2 //Bussiness Logic age > 18
3 do{
4     printf("Please Enter Your Age\n");
5     scanf("%i",&Age);
6 }while(Age<=18);
```

9

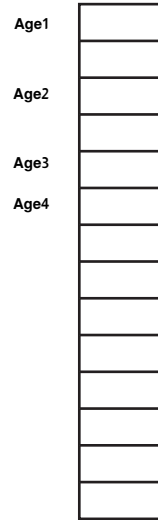
# Arrays

في حالة انك عايز تخزن مجموعة من القيم والقيم ده من نفس نوع البيانات يبقى احسن نوع بيانات لتخزينها هو المصفوفات Arrays

Very Bad

```
1 int age1 = 25;
2 int age2 = 27;
3 int age3 = 27;
4 int age4 = 27;
5 int age5 = 27;
6 int age6 = 27;
7
```

RAM --> Random Access Memory stack




## Array?

1. هي عبارة عن Fixed Numbers Of Variable
2. يعني في حالة حددت Size الخاص بالمصفوفة فهو لا يمكن تغييره
3. جميع القيم داخل المصفوفة هما من نفس نوع البيانات
4. يتم حفظ البيانات بداخل المصفوفة بشكل Sequential
5. Compile Time Data Type يعني لازم الos يكون عارف وقت ما اعمل Build هيجز مساحة كام
6. Dervid Data Type يعني اقدر اتحكم تحجز مساحة اجمالية كام في الذاكرة وقت التطوير
7. ان اسم المصفوفة نفسها هو بيمثل عنوان اول عنصر في المصفوفة.

int Age[10] --> 40Byte

```
1 //Decliration
2 int Age[5];
```

Age 

Age = 0x10

Age[0] = ##

&Age[0] = 0x10

**RAM --> Random Access Memory  
stack**

[illegible]

```
1 //Array
2     int Age[5];
3
4 //Assignment
5     Age[0]=10;
6     Age[1]=12;
7     Age[2]=144;
8     Age[3]=15;
9     Age[4]=123;
10
11
12 //Access
13     printf("%i\n",Age[0]);
14     printf("%i\n",Age[1]);
15     printf("%i\n",Age[2]);
16     printf("%i\n",Age[3]);
17     printf("%i\n",Age[4]);
```

```
1 //Decleration
2     int x = 10;
3     int y = 20;
4 //Array
5     int Age[5];
```

**RAM --> Random Access Memory  
stack**

The diagram illustrates the memory layout for the `Age` array and variables `X` and `Y`. The `Age` array is located at memory address `0x10` and contains five elements, each 4 bytes in size. The elements are `Age[0]` through `Age[4]`, each containing the value `##`. The memory addresses for these elements are `0x10`, `0x14`, `0x18`, `0x22`, and `0x26` respectively. Below the `Age` array, there are three empty 4-byte slots. Variable `X` is located at memory address `0xf12` and contains the value `10`. Variable `Y` is located at memory address `0xf9` and contains the value `20`. The memory addresses for `X` and `Y` are shown in hexadecimal. The diagram also shows a red box and a white box to the left of the `Age` array, and a blue box and a white box to the left of the `X` and `Y` variables.

Variable	Index	Value	Address
Age	Age[0]	##	0x10
	Age[1]	##	0x14
	Age[2]	##	0x18
	Age[3]	##	0x22
	Age[4]	##	0x26
Empty			
X		10	0xf12
Empty			
Y		20	0xf9
Empty			

**X**

**Y**

## For Loop →

```
1 //Array
2     int Age[5];
3
4 //Assignment
5     Age[0]=10;
6     Age[1]=12;
7     Age[2]=144;
8     Age[3]=15;
9     Age[4]=123;
10
11
12 //Access
13     printf("%i\n",Age[0]);
14     printf("%i\n",Age[1]);
15     printf("%i\n",Age[2]);
16     printf("%i\n",Age[3]);
17     printf("%i\n",Age[4]);
18     printf("\n\n_ Adresses _\n\n");
19 //Access
20     printf("%p\n",&Age);//01
21     printf("%p\n",Age);//01
22     printf("%p\n",&Age[0]);//01
23     printf("%p\n",&Age[1]);
24     printf("%p\n",&Age[2]);
25     printf("%p\n",&Age[3]);
26     printf("%p\n",&Age[4]);
```

```
1 for(int i=0;i<5;i++){
2     printf("%i\n",Age[i]);
3 }
```

Variable	Value	Address
x	10	0xf12
y	20	0xf99
Age	0x10	0x10
Age[0]	##	0x10

11

12

```

int Arr[5]={123, 6, 9, 4, 10};
int Arr[]; //False
int Arr[] = {123, 6, 9, 4, 10}; //True
int Arr[5]={123}; //123,0,0,0,0;
int Arr[5]; //### True
int Arr[5]={}; //0,0,0,0,0 //True
int Arr = {1,2,3,4}; //False

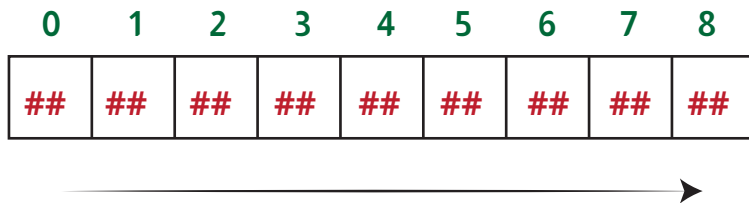
```

```

1 int Arr[5]={123, 6, 9, 4, 10};
2 int Arr[]; //False
3 int Arr[] = {123, 6, 9, 4, 10}; //True
4 int Arr[5]={123}; //123,0,0,0,0;
5 int Arr[5]; //### True
6 int Arr[5]={}; //0,0,0,0,0 //True
7 int Arr = {1,2,3,4}; //False

```

## Array 1D



## Array 2D

```

1 int Arr[5][5];

```

```

1 Arr[0][1];
2
3 Arr[0][2];
4
5 Arr[1][1];
6
7 Arr[3][1];

```

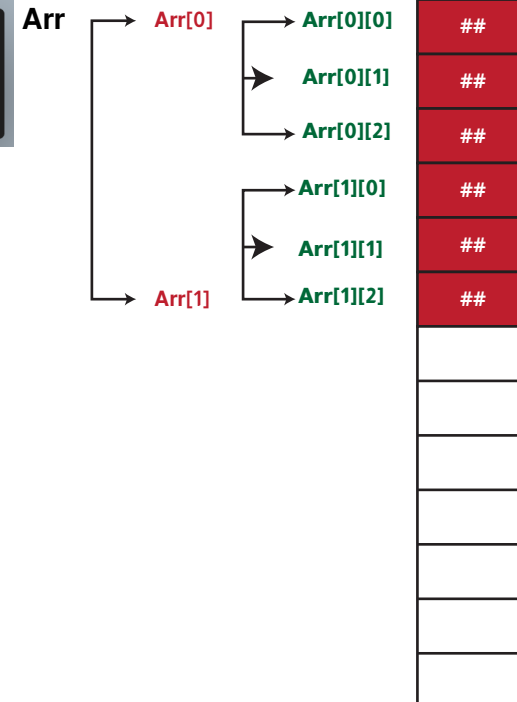


RAM --> Random Access Memory stack

```

1 int Arr[2][3]

```



String --> Next Session

A	L	I
M	A	Y

