# Kiểu con trỏ

**Khai báo :**

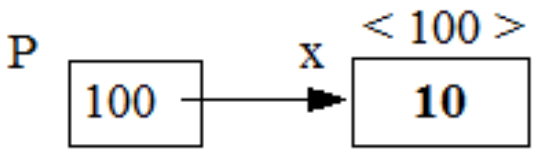
**T*  P;**

- Giá trị chứa trong P là *địa chỉ* của vùng nhớ chứa giá trị có kiểu **T**,
- Ta cũng nói P là con trỏ trỏ tới vùng nhớ chứa giá trị có kiểu **T**, hay P là con trỏ có kiểu **T**.
- Giá trị NULL : Khi P có giá trị này thì xem như P không trỏ vào đâu cả. Câu lệnh thường dùng

**if (P!=NULL) { . . . }**

VD :

**1.** int  *P;
**2.** int x;
**3.** x = 10;
**4.** P = **&**x ;



## 5.2 Truy xuất vùng nhớ trỏ bởi con trỏ:

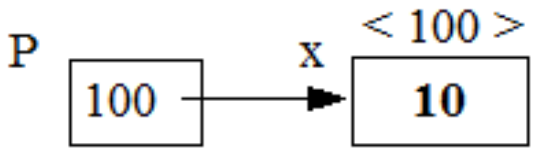
Cho khai báo :

**T*  P ;**

- **Vùng nhớ được trỏ bởi P là *P.**

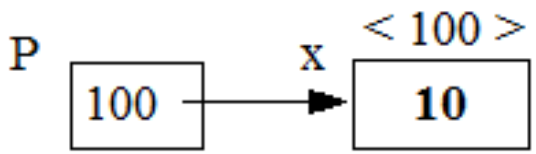
VD :

```
int*   P; // là con trỏ có kiểu int
int x;
x = 10;
P = &x ;
prinf("%d", *P);
```



VD :

```
int*   P; // là con trỏ có kiểu int
int x;
x = 10;
P = &x ;
*P = *P + 2;
prinf("%d", x);
```



Lệnh **scanf(“—”, &x)** : nhập giá trị vào vùng nhớ có địa chỉ là địa chỉ của x, tức là nhập giá trị cho biến x.

VD : Đoạn chương trình sau :

```
int x;
scanf(“%d”, &x);
printf(“%d”, x);
```

tương đương với

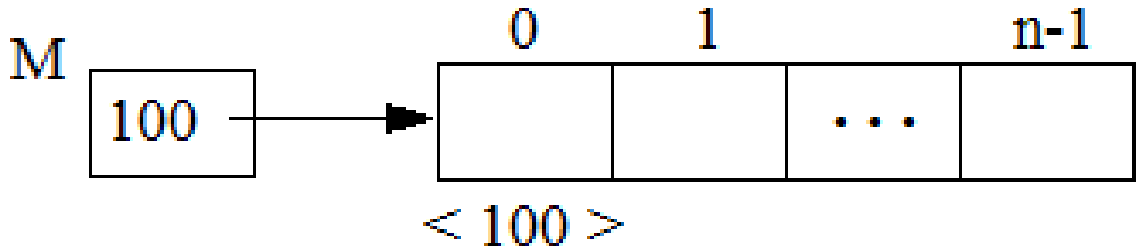
```
int x, *P;
P=&x;
scanf(“%d”, P);
printf(“%d”, x);
```

## 5.3 Mảng và con trỏ:

Cho khai báo mảng :

**T M[n] ;**

- Ta có một mảng n phần tử kiểu T.
- **M là con trỏ trỏ tới phần tử đầu tiên của mảng** (chứa địa chỉ của phần tử đầu tiên của mảng).



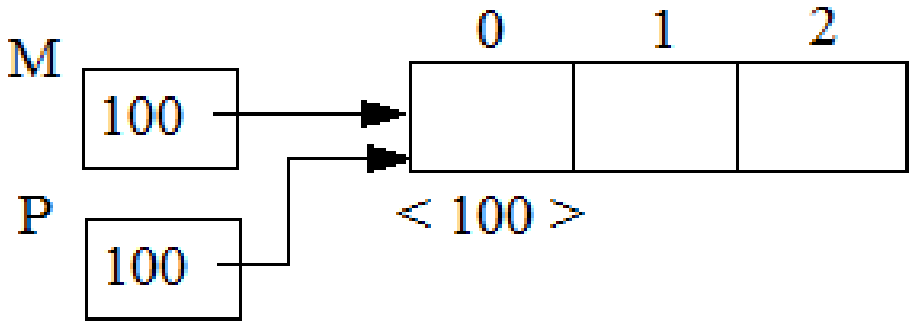

- (M + i) , i = 0, . . ., n-1, là địa chỉ của phần tử thứ i của mảng. Vậy *(M+i) là M[i].
- Mảng được khai báo như trên còn được gọi là *mảng tĩnh*.

VD 1:

```
int  M[3];
*(M+0)=10; // ⇔ *M=10 ; ⇔ M[0]=10 ;
*(M+1)=20; // ⇔  M[1]=20 ;
*(M+2)=30; // ⇔  M[2]=20 ;
```

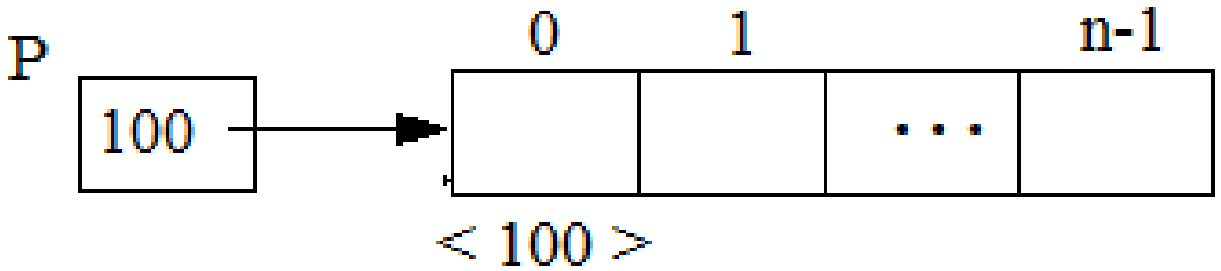
VD 2:

```
int  M[3], *P;
P=M;
*(P+0)=10; // ⇔ P[0]=10; ⇔  M[0]=10 ;
*(P+1)=20; // ⇔ P[1]=20; ⇔  M[1]=20 ;
*(P+2)=30; // ⇔ P[2]=20; ⇔  M[2]=20 ;
```



## 5.4 Cấp phát vùng nhớ cho con trỏ :

```
#include <malloc.h>
int main(. . .)
{
        T *P;
        P=(T*)malloc(n*sizeof(T)); (1)
        . . .
}
```




- (1) sẽ cấp phát **n** ô nhớ có kiểu là T và con trỏ P sẽ trỏ tới ô nhớ đầu tiên,
- P là mảng kiểu T, được gọi là *mảng động*, các phần tử của mảng là P[i], i=0, . ., n-1,
- Nếu không cấp phát được thì P có giá trị NULL.

**Ví dụ :**

```
#include <malloc.h>
int main(int argc, char* argv[])
{
        int *P;
        P=(int*)malloc(1*sizeof(int)); // ⇔ P=(int*)malloc(sizeof(int));
        *P=10; // P[0]=10;
        printf("*P= %d\n",*P);

        return 0;
}
```

**Ví dụ :**

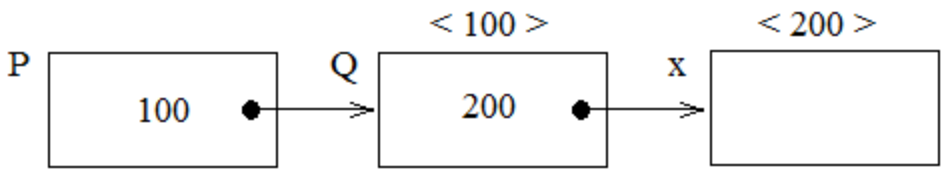
```
#include <malloc.h>
int main(int argc, char* argv[])
{
        int *P;
        P=(int*)malloc(3*sizeof(int));
        P[0]=10; P[1]=100; P[2]=1000;
        printf("%d, %d, %d",P[0], P[1], P[2]);
    // printf("%d, %d, %d",*P, *(P+1), *(P+2));
        return 0;
}
```

## 5.5 con trỏ có kiểu con trỏ (kiểu T) :

## 5.5.1 Khai báo con trỏ có kiểu con trỏ :

**T **P;**

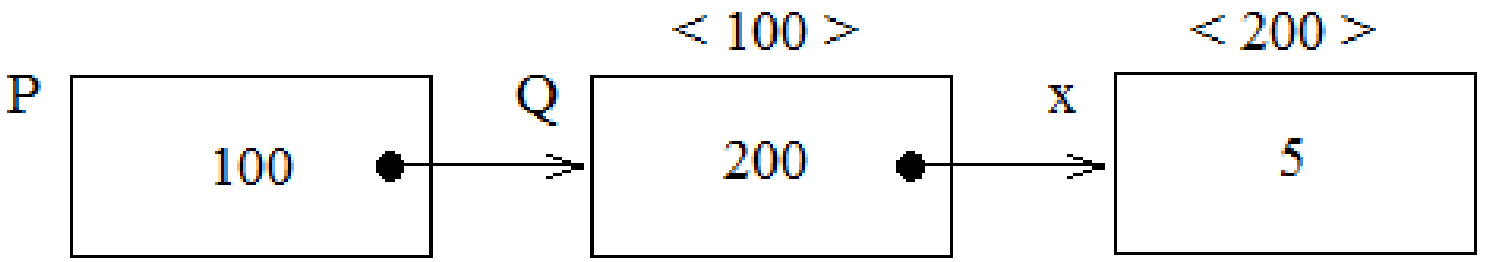
- P là con trỏ (chứa địa chỉ) trỏ tới vùng nhớ, (vùng nhớ này) là con trỏ trỏ tới vùng nhớ chứa giá trị có kiểu T.




- *P là vùng nhớ được trỏ bởi P, gọi vùng nhớ này là Q, thì **P là *Q, là vùng nhớ kiểu T.

Ví dụ :

```
int x , *Q, **P;
x = 5;
Q = &x;
P = &Q;
printf("%d\n", **P);
```




Ví dụ :

```
int x , *Q, **P;
x = 5;
Q = &x;
P = &Q;
**P = **P +1;
printf("%d\n", x);
```

## 5.5.2 Cấp phát vùng nhớ cho *Con trỏ có kiểu con trỏ* (kiểu T) :

Cho khai báo :

**T **P;**

P=(T**)malloc(**m***sizeof(T*)); (1)  P[i] : con trỏ , trỏ tới T

P[i] = (T*)malloc(**$n_i$***sizeof(T)); // i = 0, . . ., m-1  (2)

- (1) : P là con trỏ, trỏ tới phần tử đầu tiên của mảng **m** phần tử. Mỗi phần tử trong mảng có kiểu *con trỏ kiểu T*, P[i], i =0, . . ., m-1
- (2) : Cấp phát mảng $n_i$ phần tử kiểu T cho P[i] (P[i] trỏ tới phần tử đầu tiên của mảng), P[i][j], j =0, . . ., $n_i$-1

```
T  **P;
P=(T**)malloc(m*sizeof(T*)); (1)
P[i] = (T*)malloc(n_i*sizeof(T)); // i = 0, . . ., m-1 (2)
```

**Ví dụ**

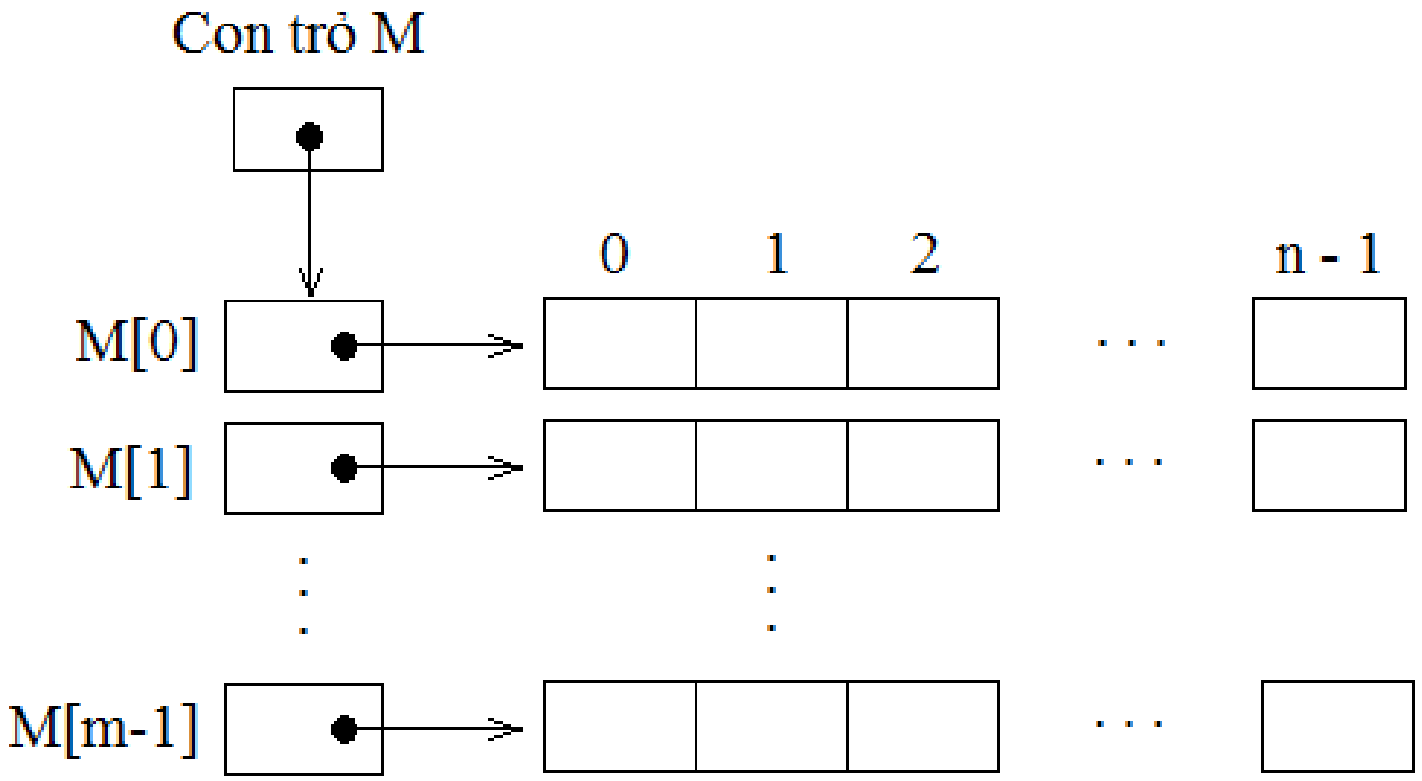
```
int  **P;
P=(int**)malloc(2*sizeof(int*));    (1)
P[0] = (int*)malloc(2*sizeof(int)); // Ta có : P[0][0], P[0][1] kiểu int      (2)
P[1] = (int*)malloc(3*sizeof(int)); // Ta có : P[1][0], P[1][1], P[1][2]    (2)

*(P[1]+1)=10; // ⇔ P[1][1] = 10;
printf("kq = %d\n",P[1][1]);
```

## 5.5.3 Con trỏ và mảng 2 chiều :

Cho khai báo :

**T M[m][n];**

M là con trỏ có kiểu con trỏ (có kiểu T).



## 5.6 Mảng các con trỏ kiểu T :

Cho khai báo : T **P, *M[n], A[m][n];

**T *M[m];** // (**T **P;** P=(int**)malloc(**m***sizeof(int*));)

- M[i], i=0, 1 , . . ., m-1, là con trỏ kiểu T,
- M[i] = (T*)malloc(**$n_i$***sizeof(T)); // i = 0, . . ., n-1

VD :

```
int *M[2];
M[0] = (int*)malloc(2*sizeof(int)); // ⇔ M[0][0], M[0][1]
M[1] = (int*)malloc(3*sizeof(int)); // ⇔ M[1][0], M[1][1], M[1][2]
*(M[1]+1)=10;
printf("kq = %d\n",M[1][1]);
```

## 5.7 Giải phóng vùng nhớ cấp phát cho p :

```
T *p;
p=(T*)malloc(N*sizeof(T));
. . .
free(p);
```

**Chuỗi và con trỏ :**

VD :

```
char  S[30], *P;
strcpy(S, “ABCDEFGH”);
P = (S+2);
printf("%s\n", P); // Trên màn màn hình là : CDEFGH
```

VD :

```
char  S[30], *P;
strcpy(S, "ABCDEFGH");
P = (S+2);
strcpy(P, "1234");
printf("%s\n", S); // Trên màn màn hình là : ?
```