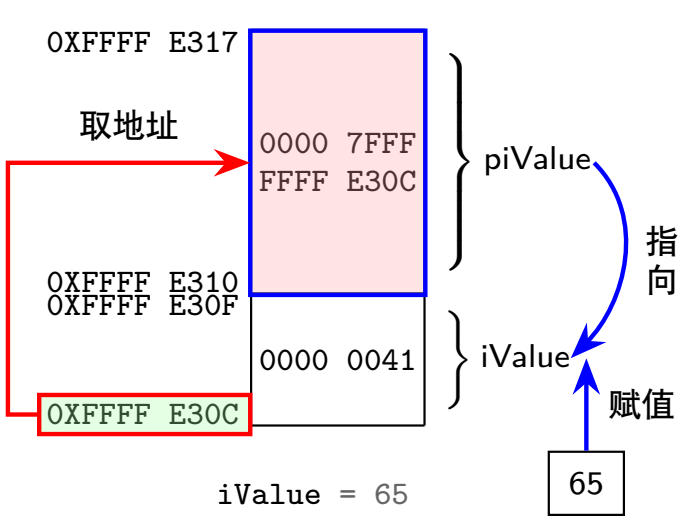
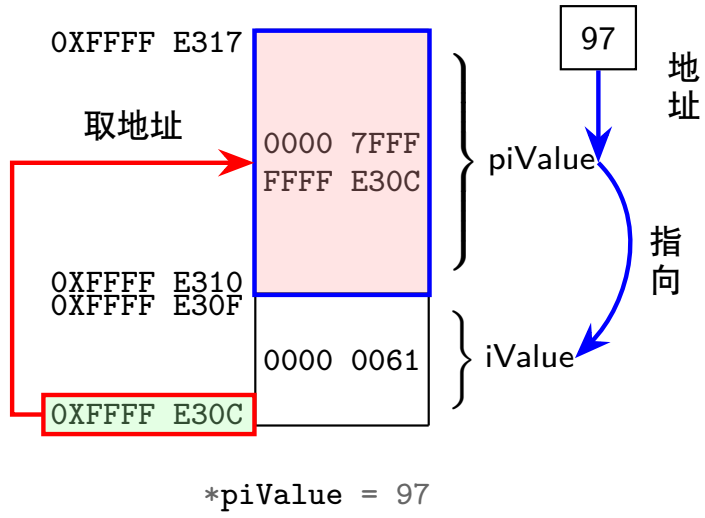
应用实例

```
int iValue;          /* 整型类型 */
int * piValue;       /* 整型地址类型 */

piValue = &iValue;   /* 将 iValue 的地址赋与指针 */
iValue = 65;         /* 给 iValue 赋值 */
*piValue = 97;       /* 给 piValue 指向的内存赋值 */
```



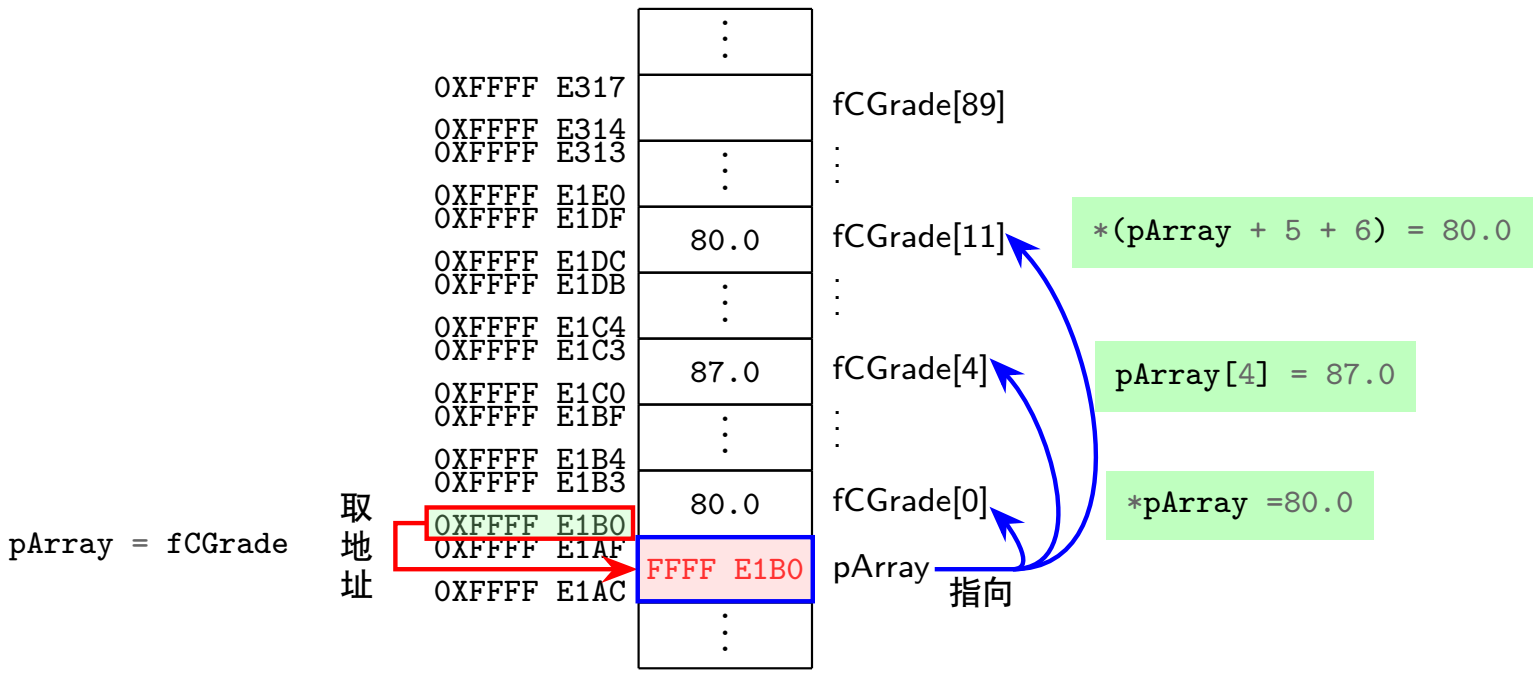


此处省略 64 位地址的前 4 个字节 0X0000 7FFF。

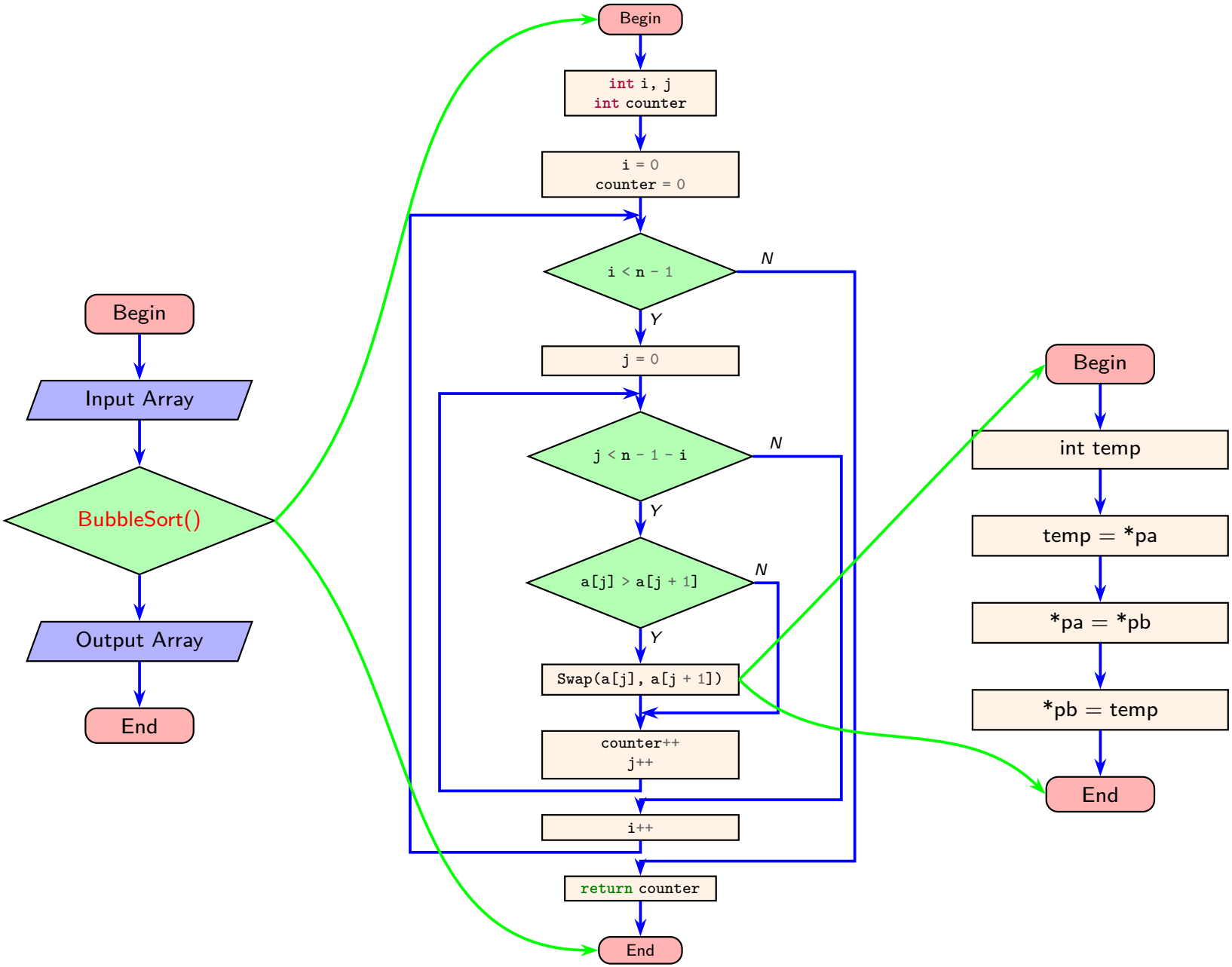
同种数据类型的数据集合

指针索引

```
int i = 0;
float fCGrade[90];          /* 声明数组 */
float * pArray;             /* 声明指针，注意是浮点型地址类型的指针 */
pArray = fCGrade;           /* 将数组首地址赋给指针 */
*pArray = 80.0;             /* 相当于 fCGrade[0]*/
*(pArray + 5 + 6) = 80.0; /* 相当于 fCGrade[11]*/
pArray[4] = 87.0;           /* 相当于 fCGrade[4]*/
pArray = &fCGrade[5];       /* 将 fCGrade[5] 的地址赋给指针 */
pArray[4] = 79.0;           /* 相当于 fCGrade[9]*/
```

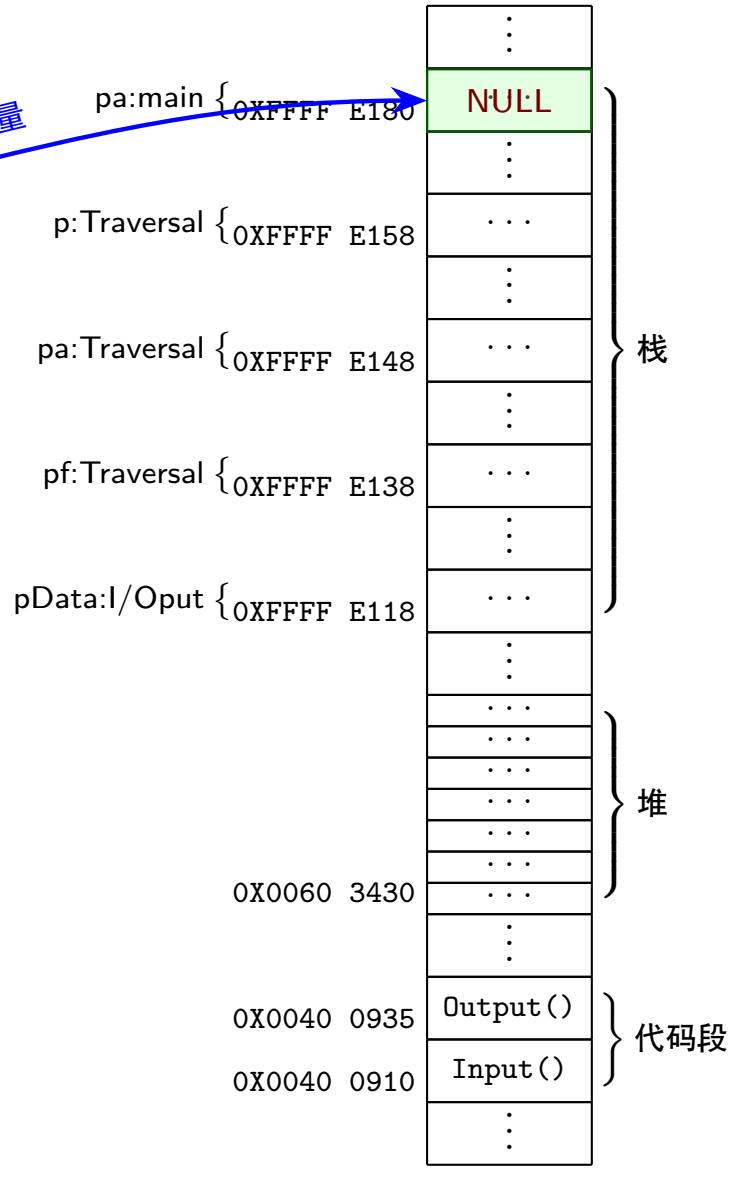


冒泡排序 (减少循环次数)


Begin
Input Array
BubbleSort()
Output Array
End
Begin
int i, j
int counter
i = 0
counter = 0
i < n - 1
N
Y
j = 0
j < n - 1 - i
N
Y
a[j] > a[j + 1]
N
Y
Swap(a[j], a[j + 1])
counter++
j++
i++
return counter
End
Begin
int temp
temp = *pa
*pa = *pb
*pb = temp
End

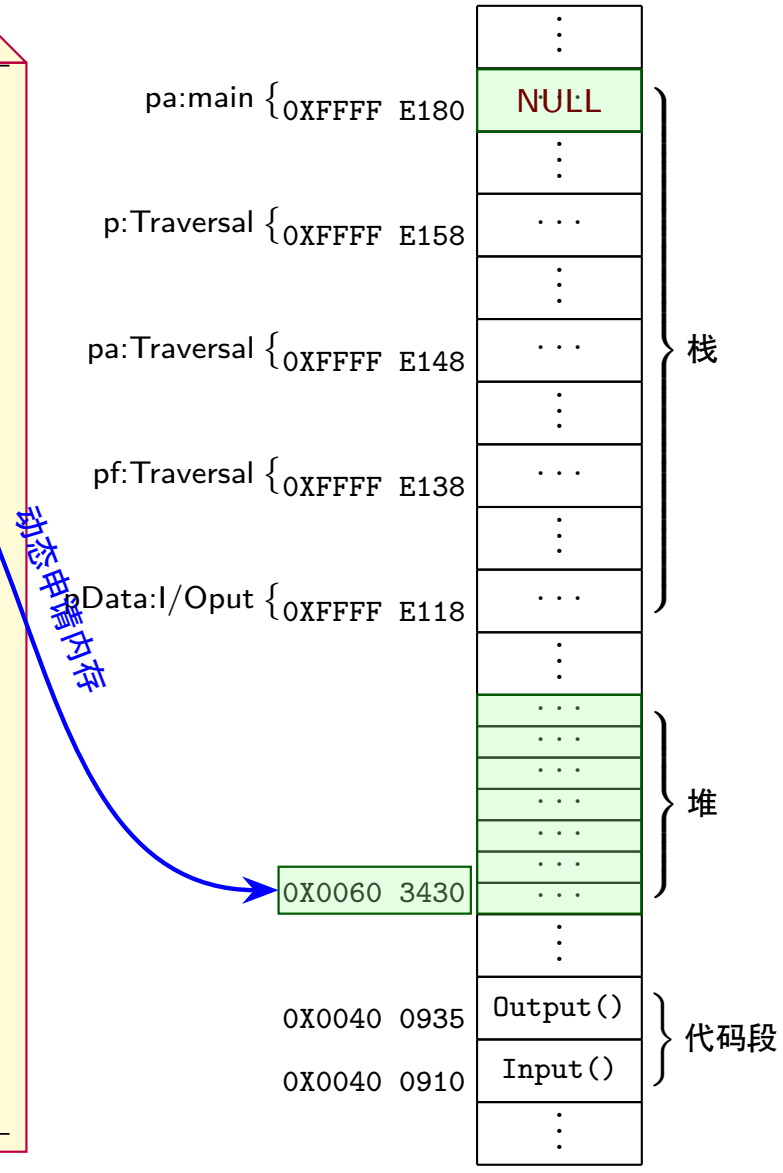
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



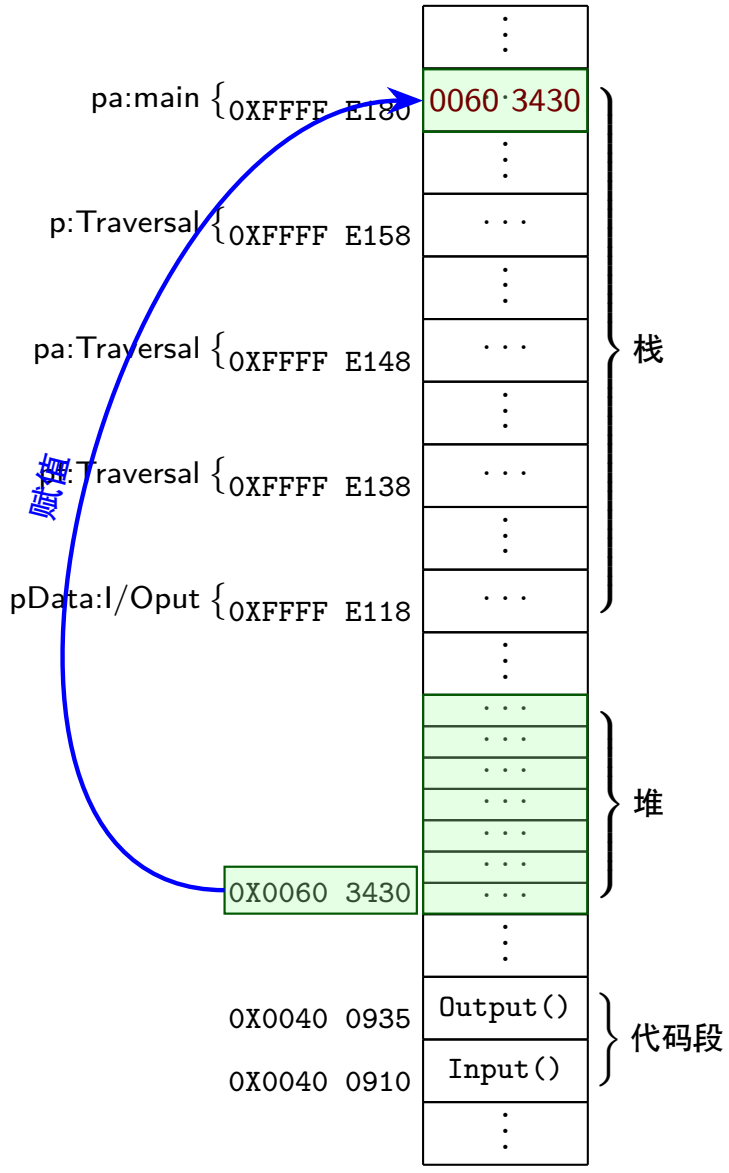
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



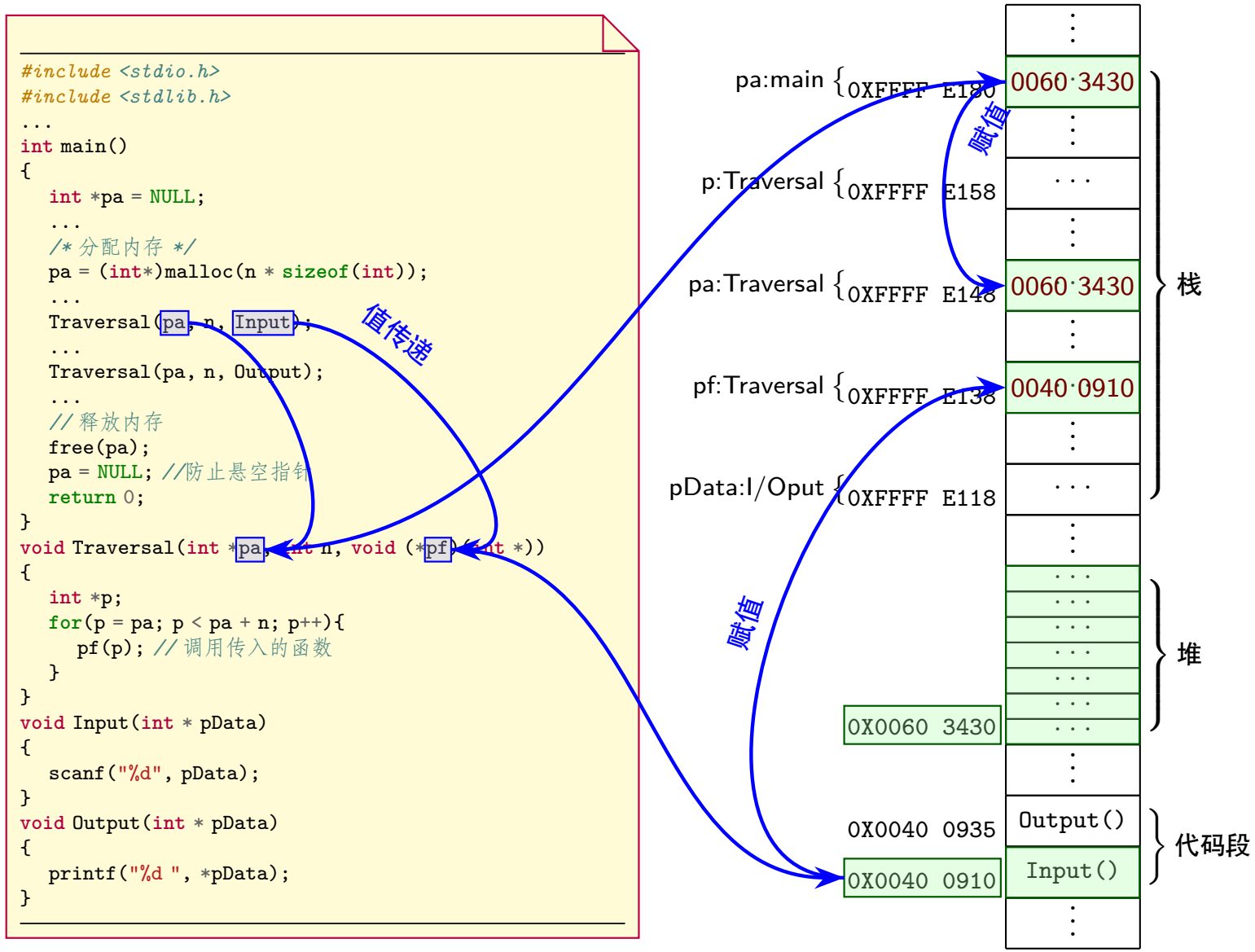
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



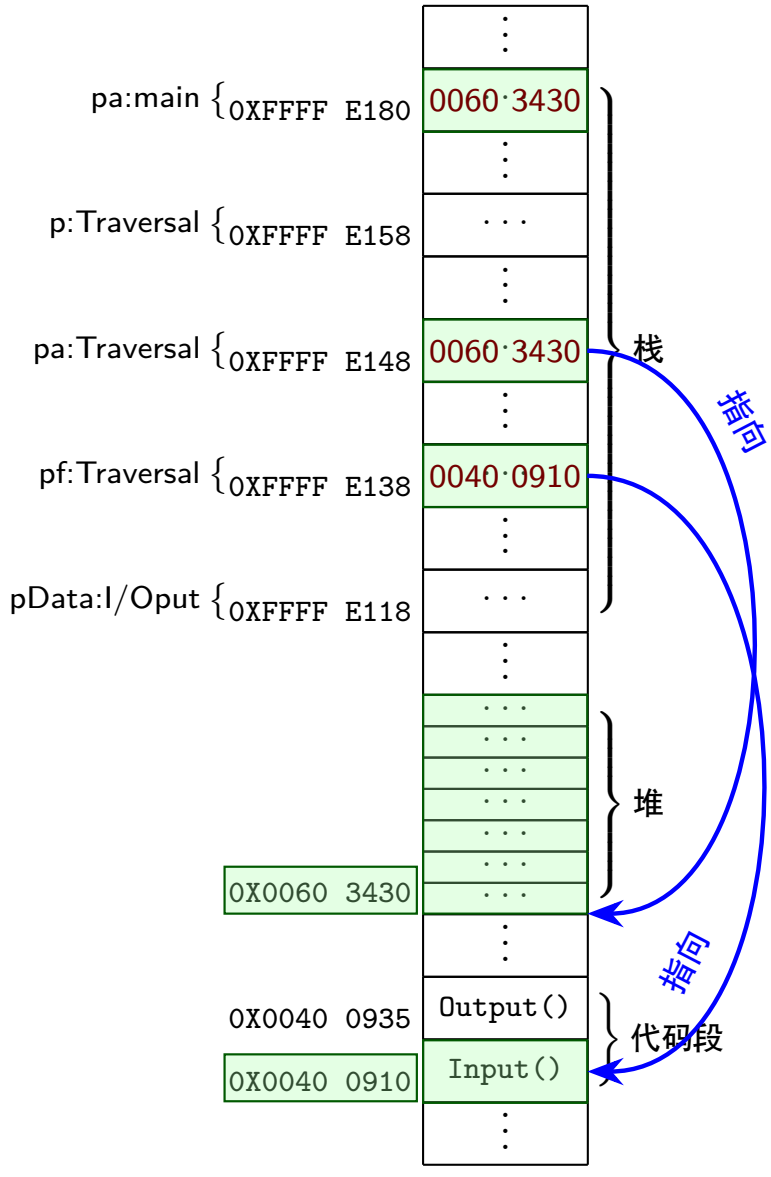
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



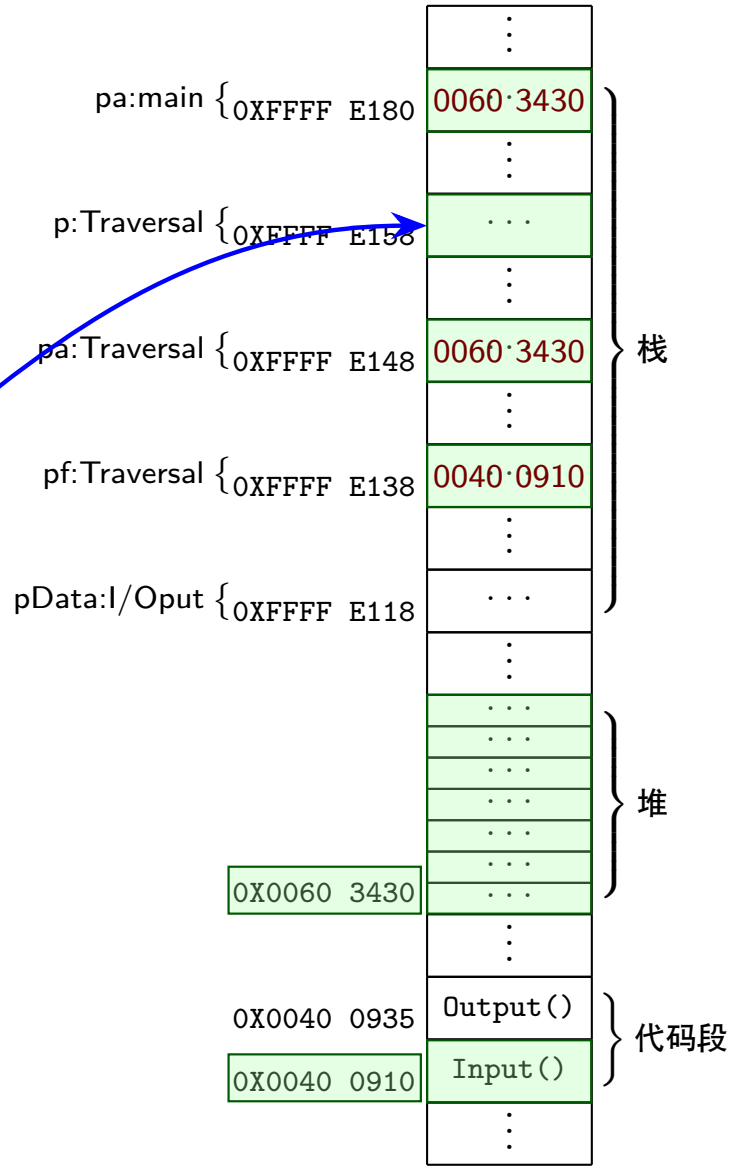
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



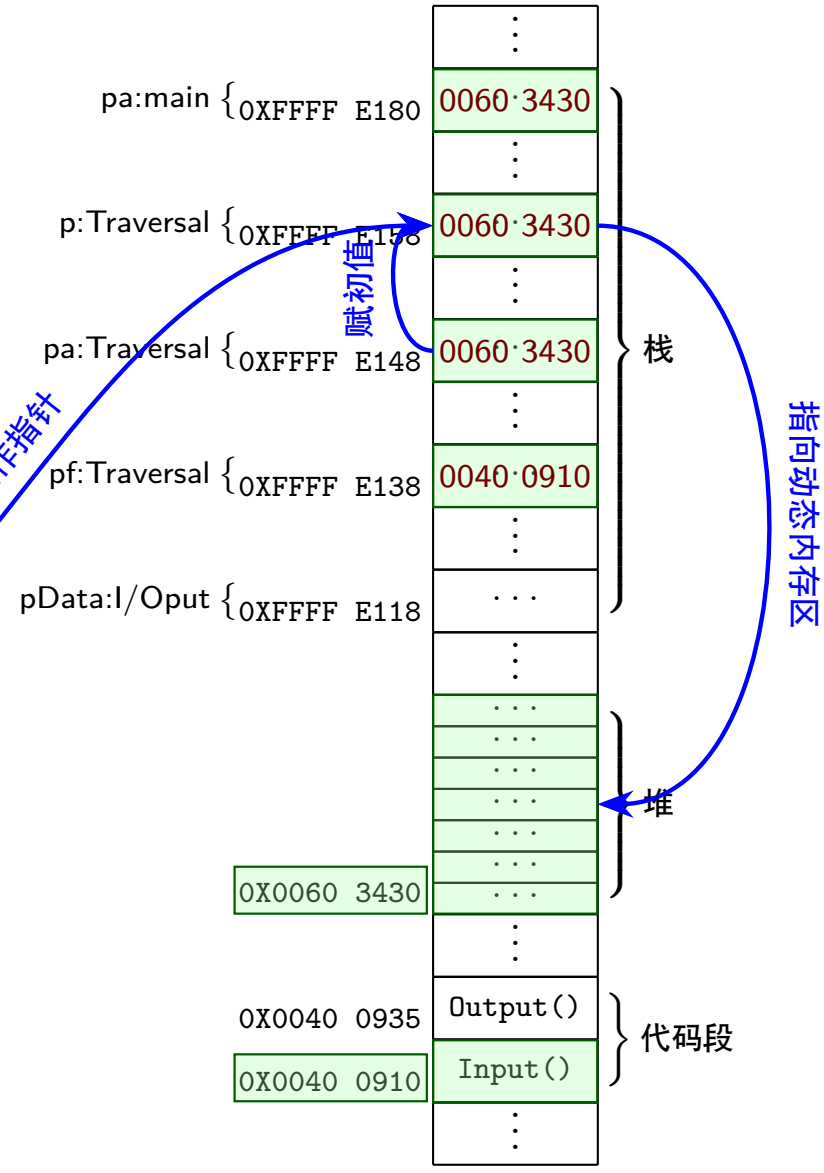
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



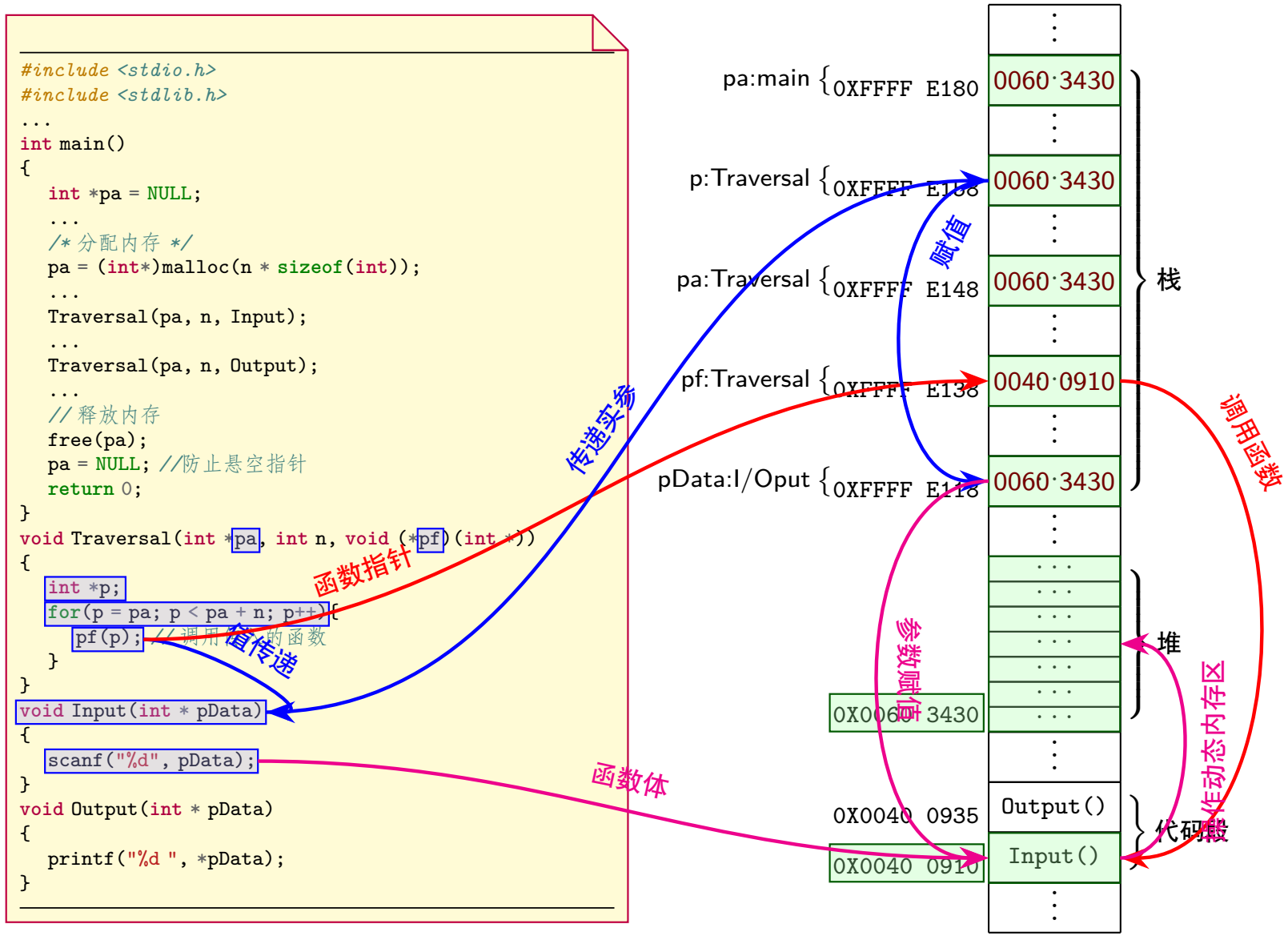
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



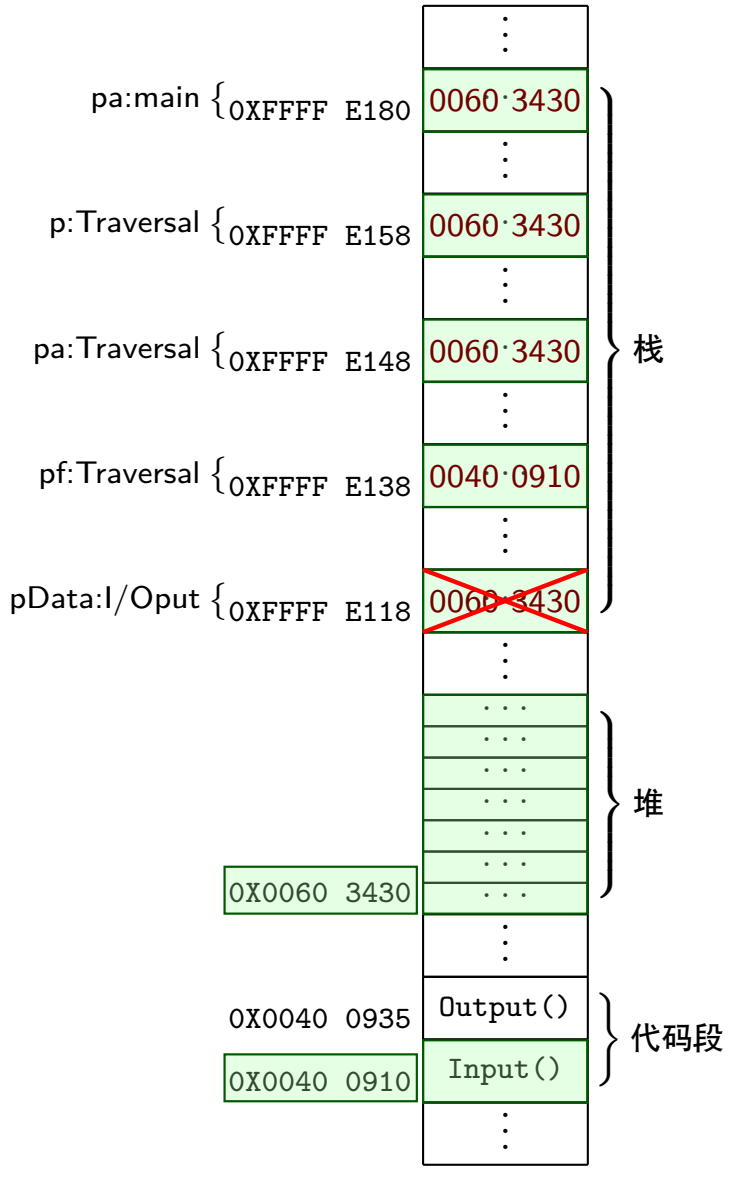
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); //调用参数的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



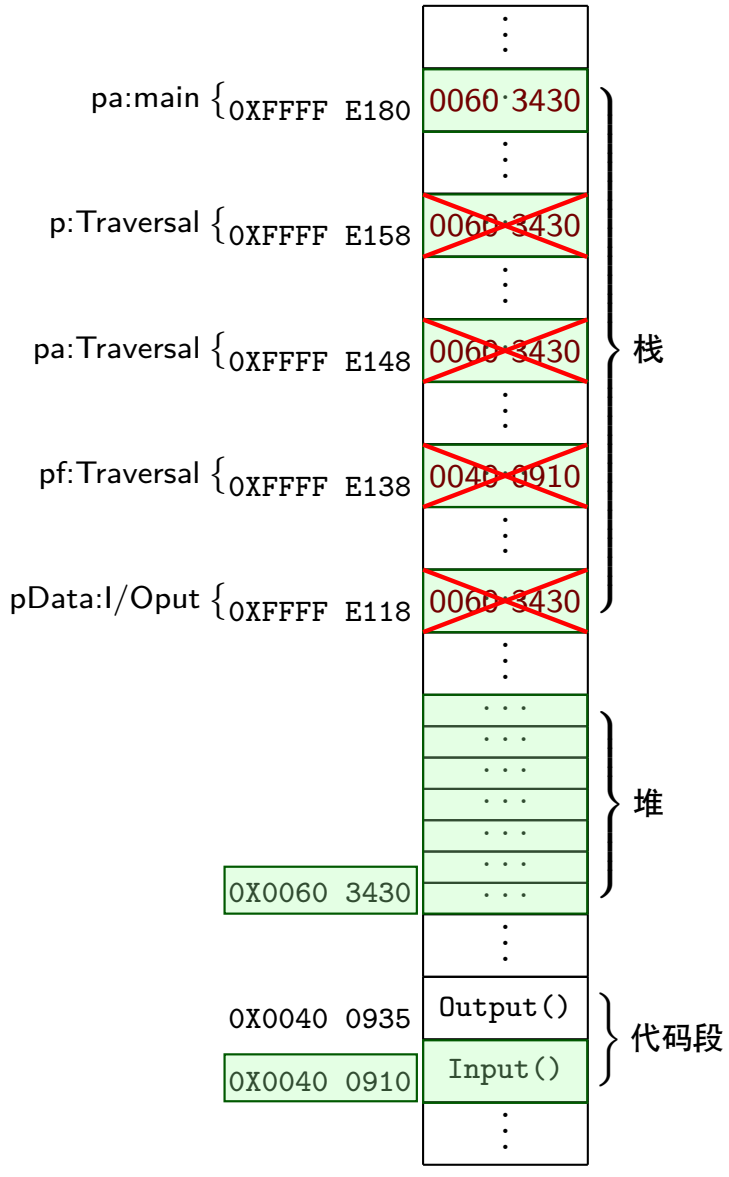
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



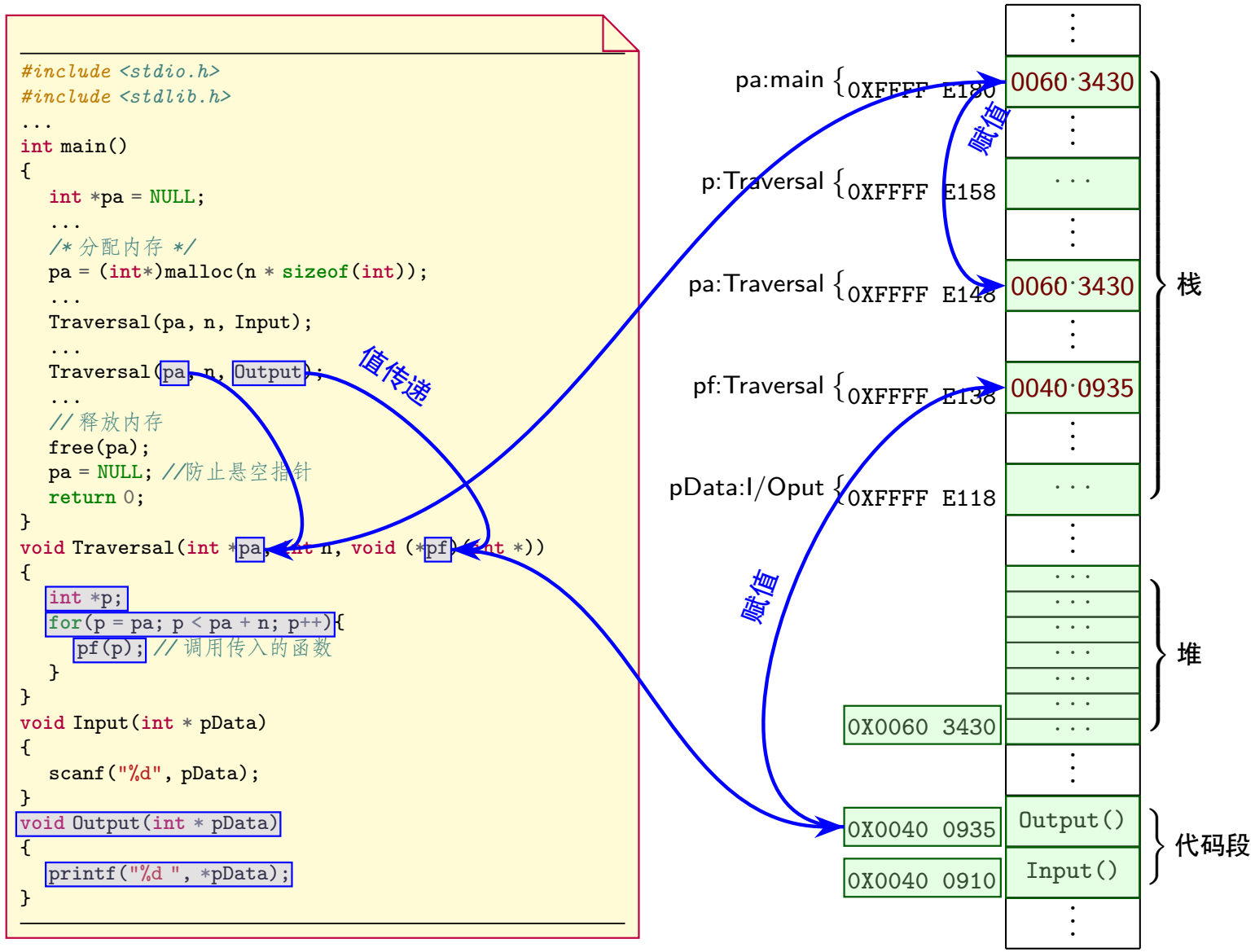
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



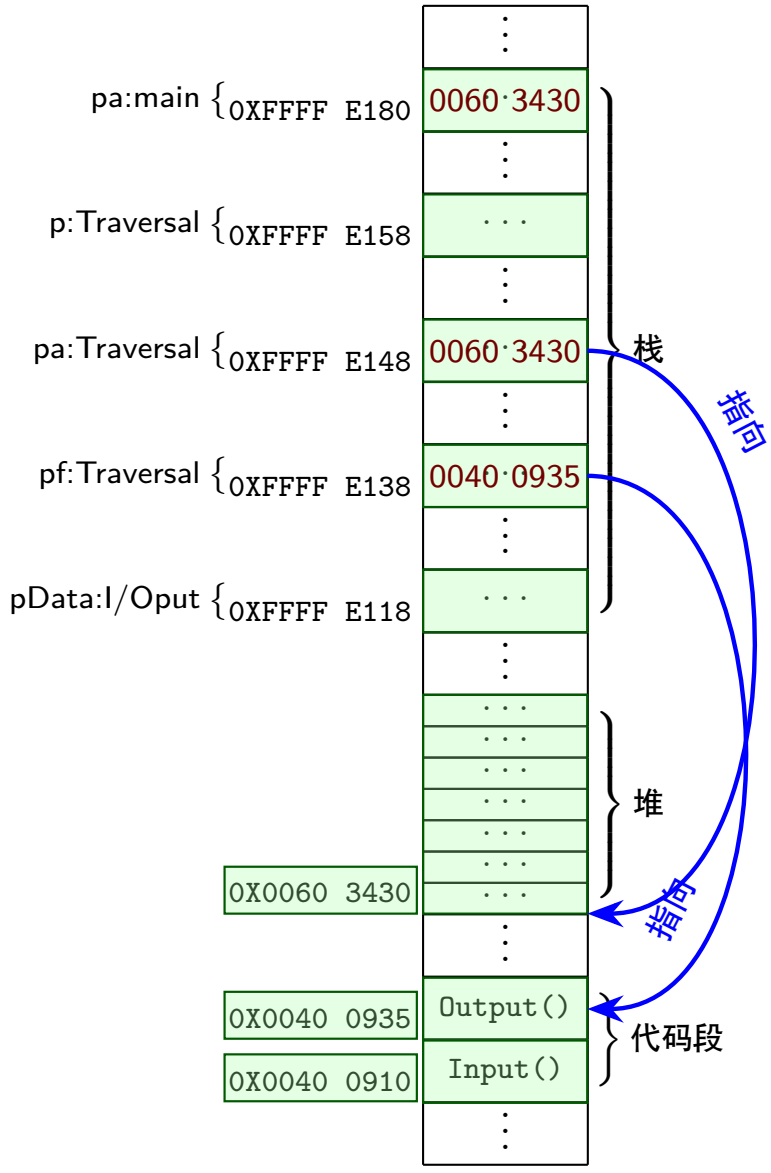
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



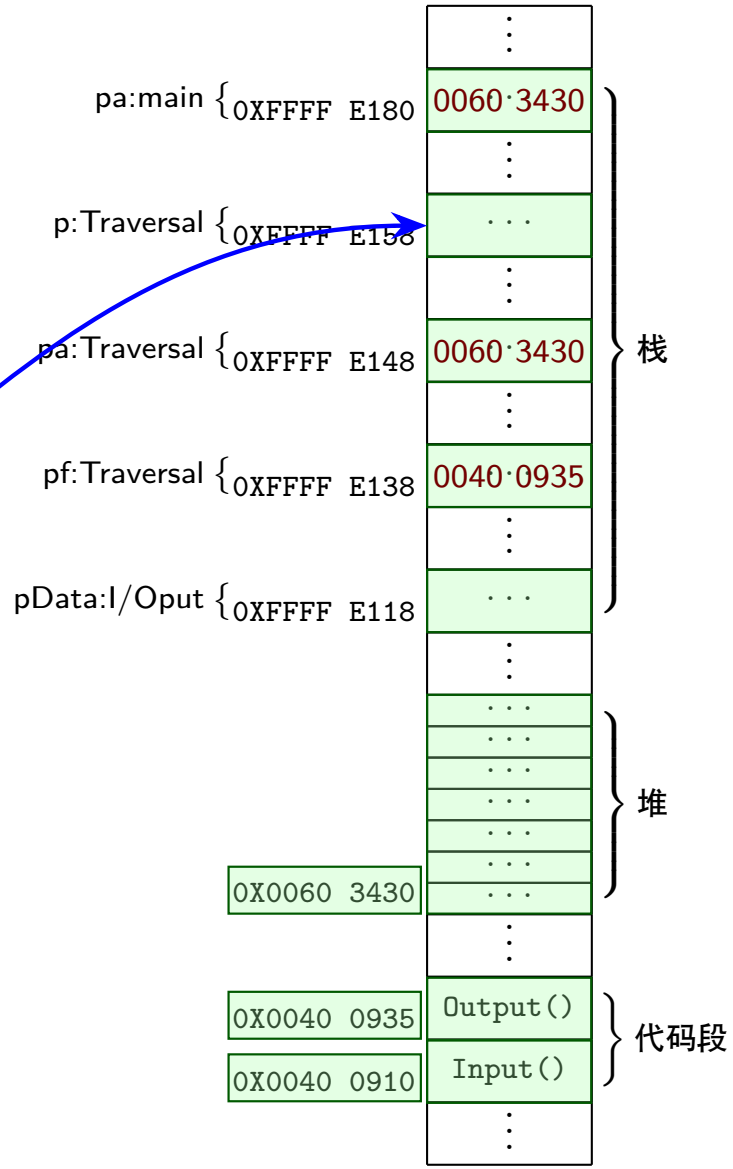
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



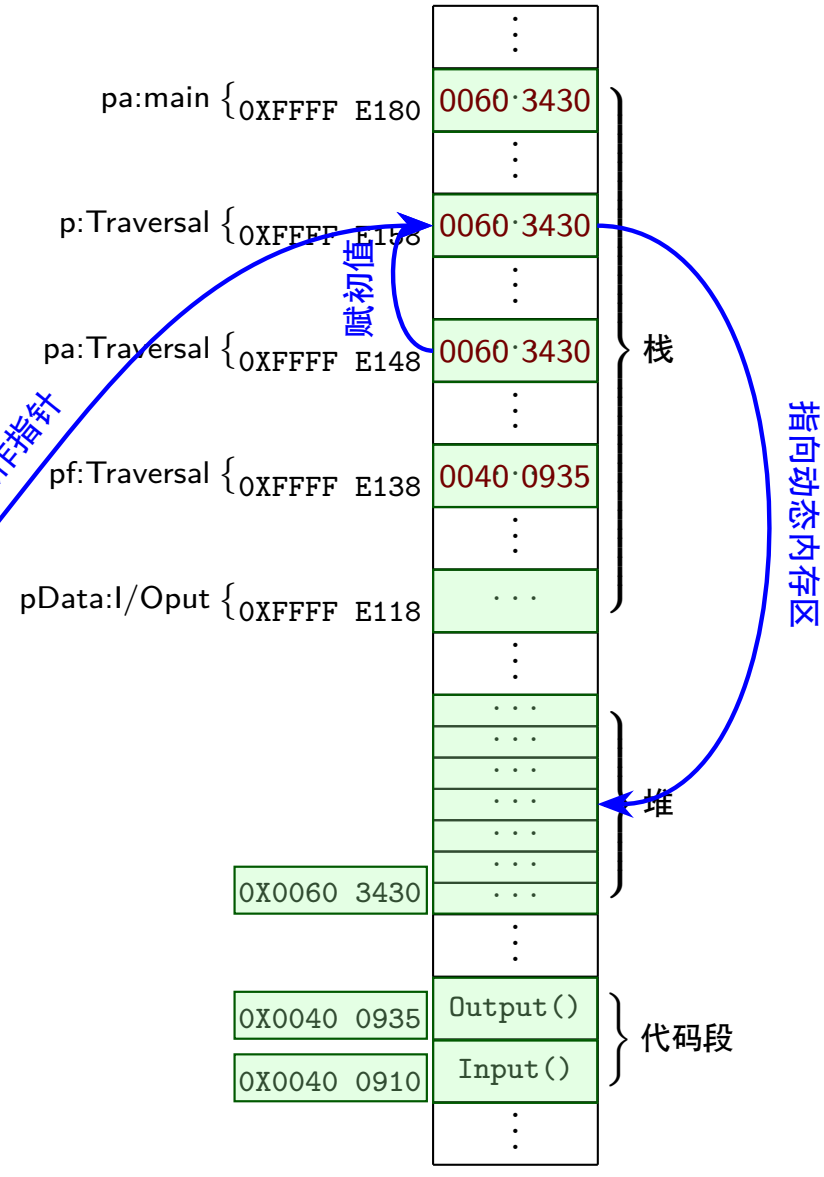
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



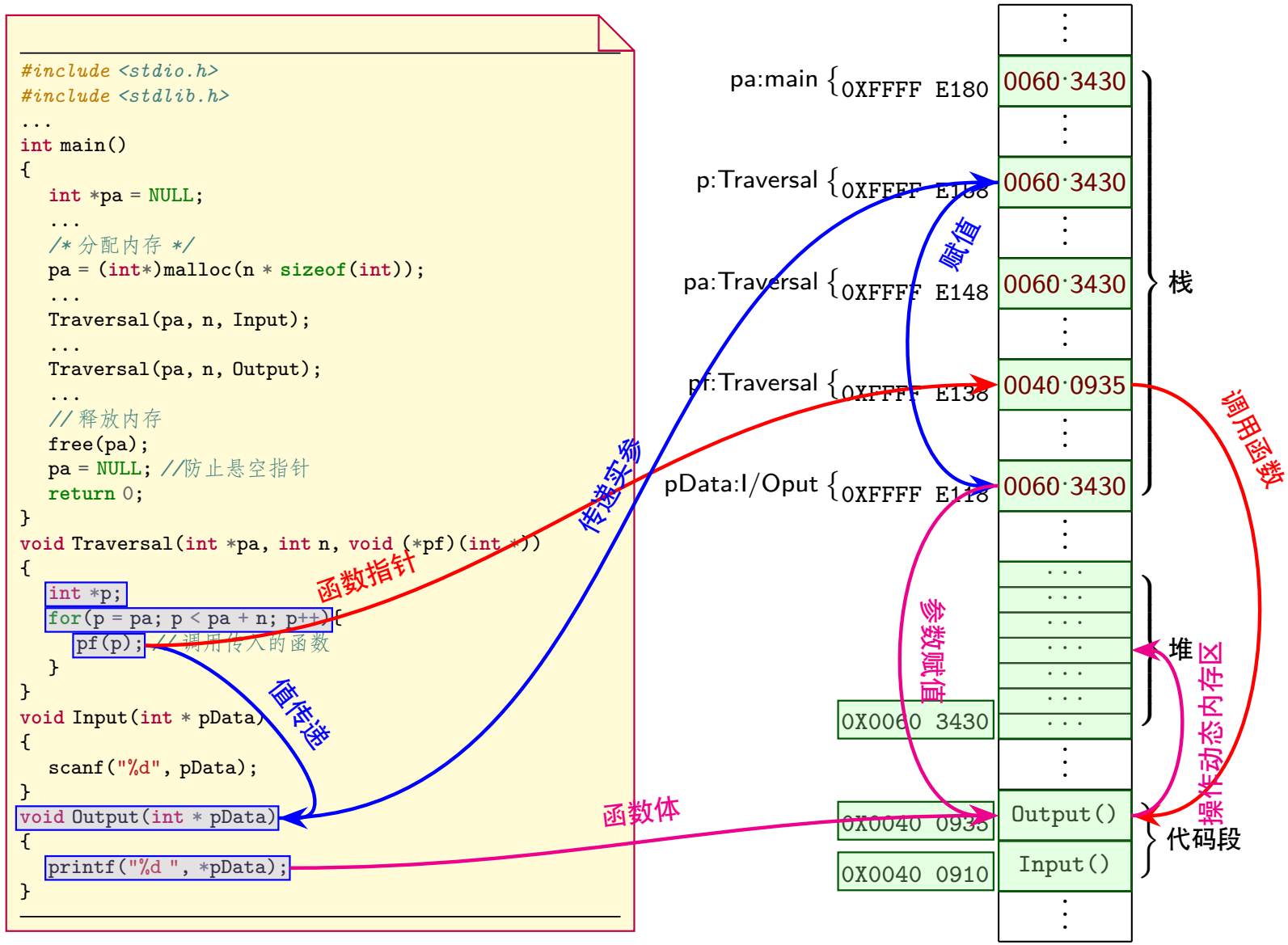
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



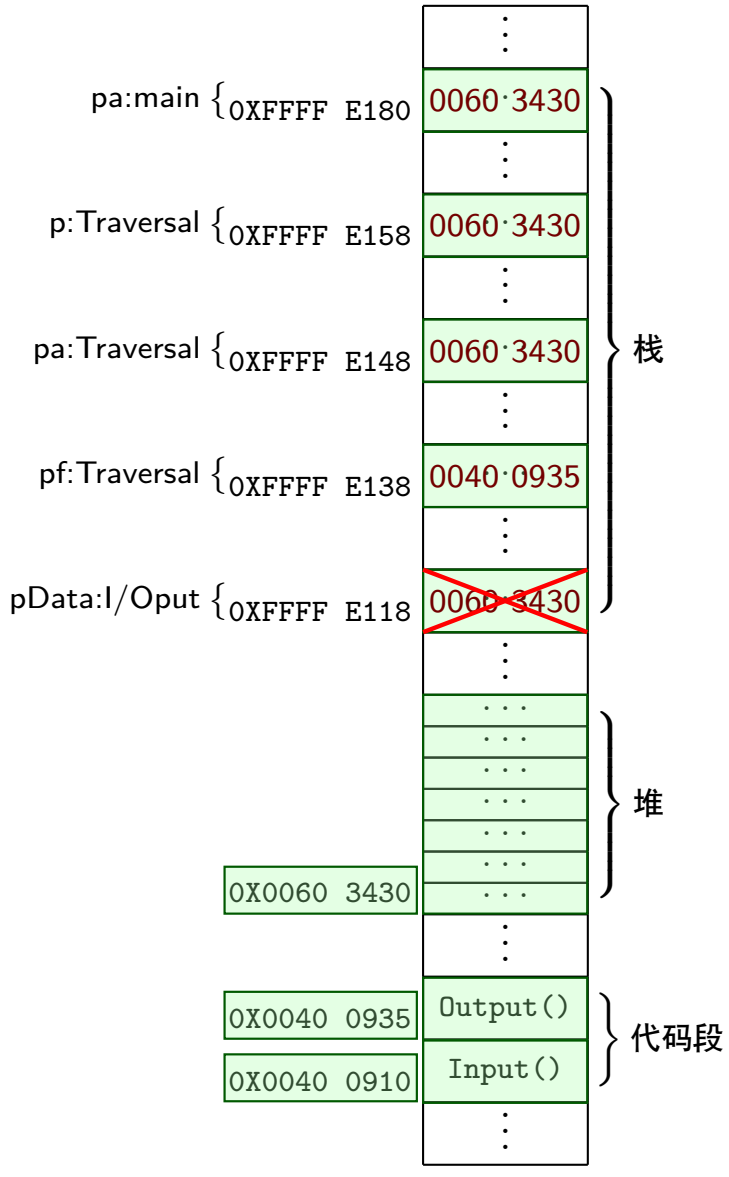
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); //调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



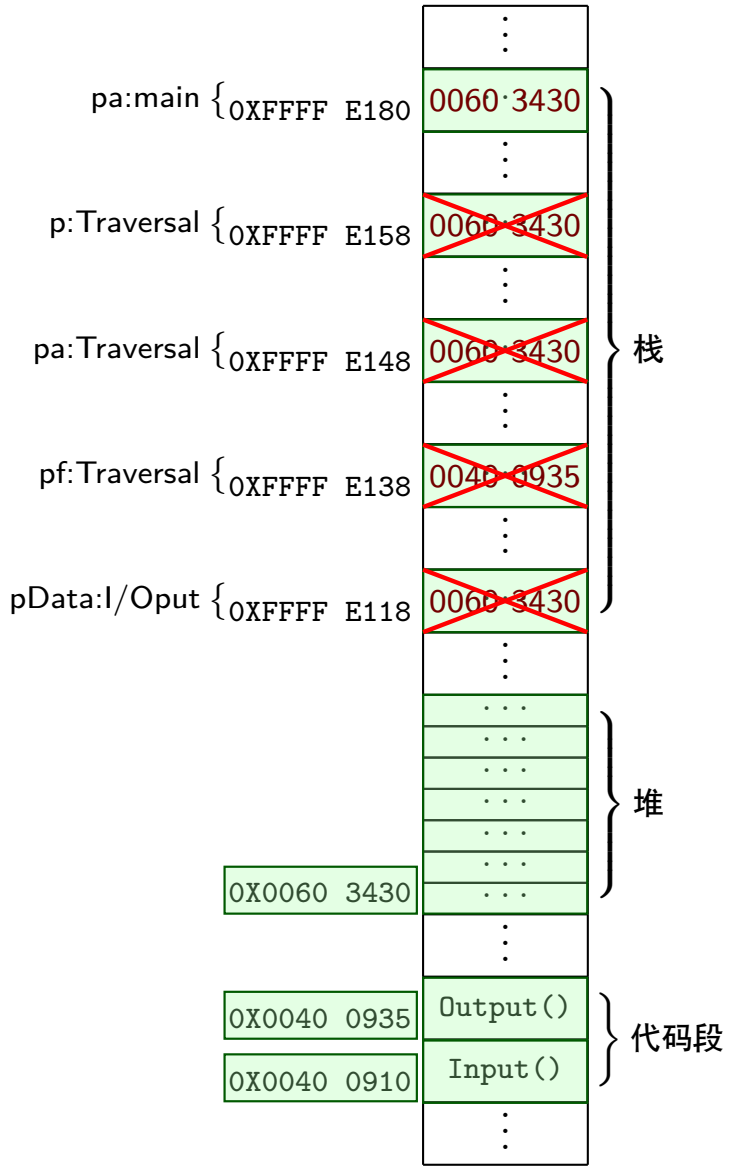
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



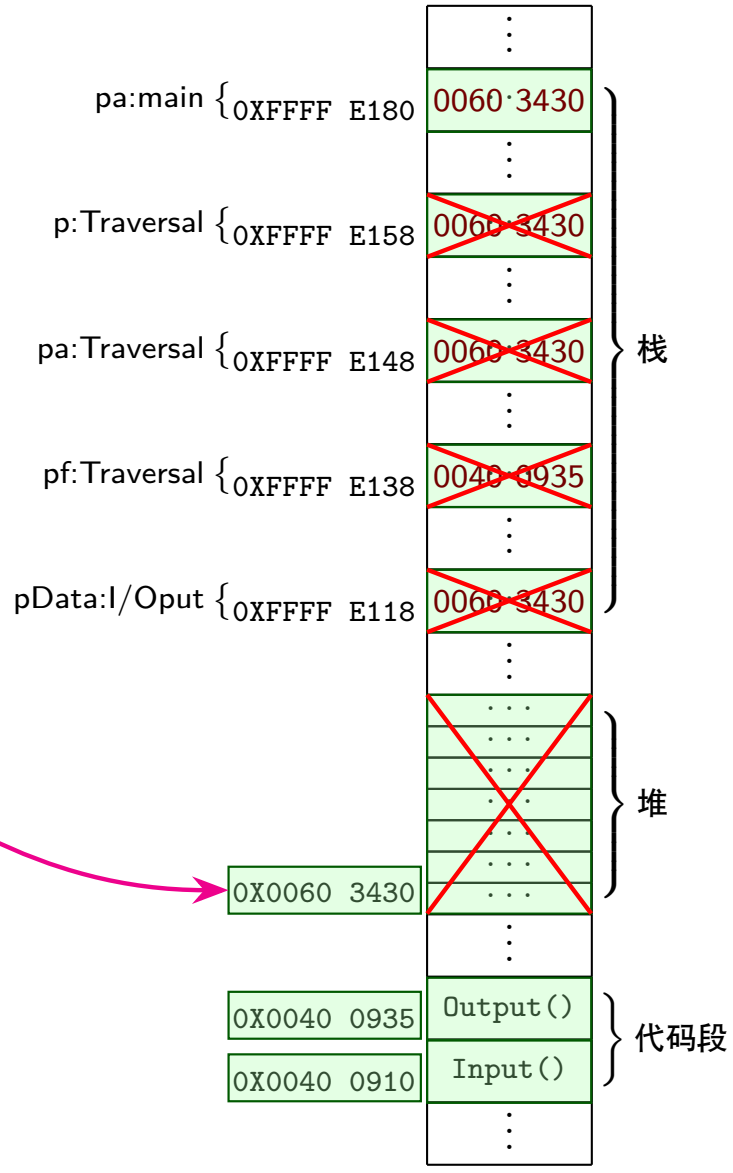
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



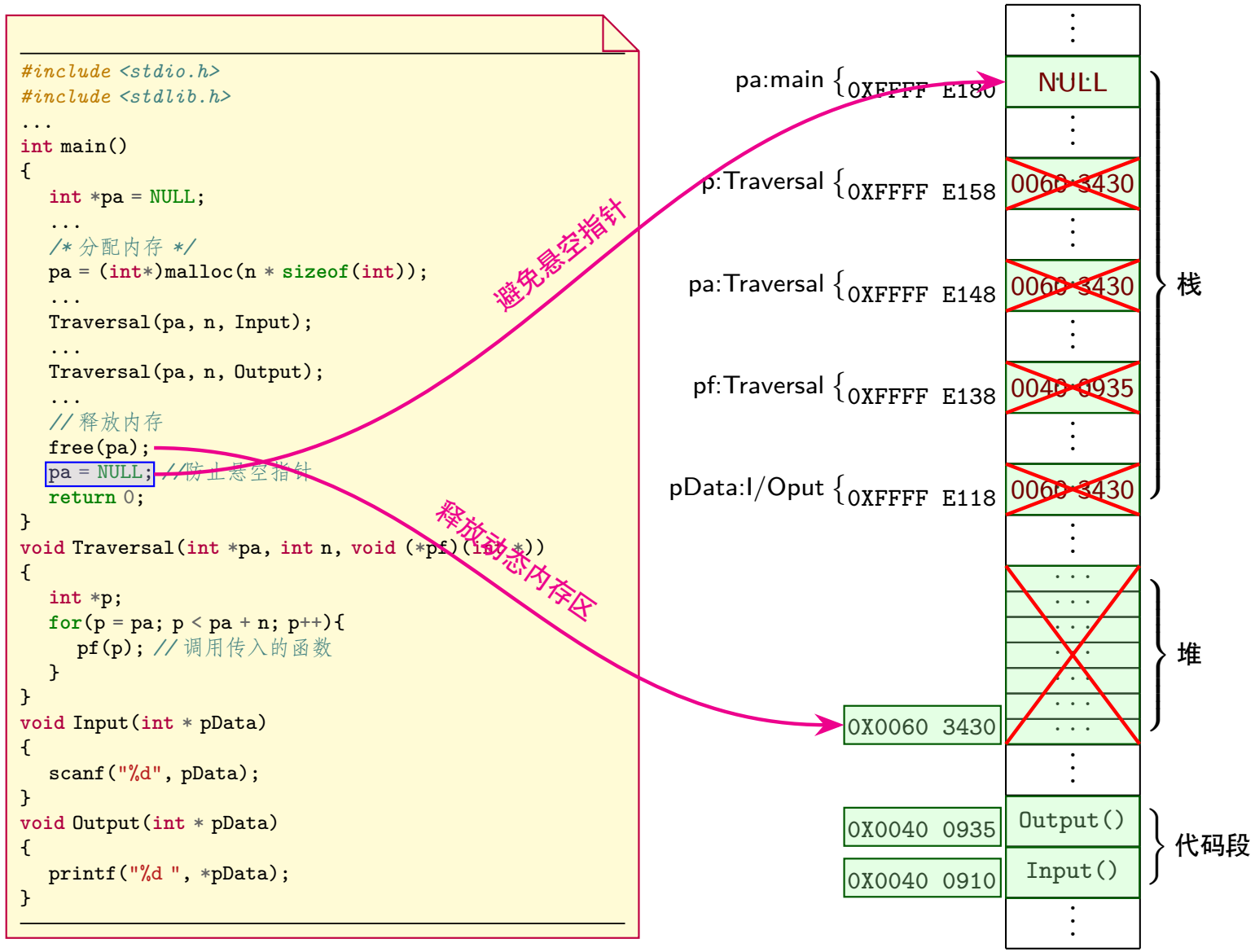
动态数组作函数参数

```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```

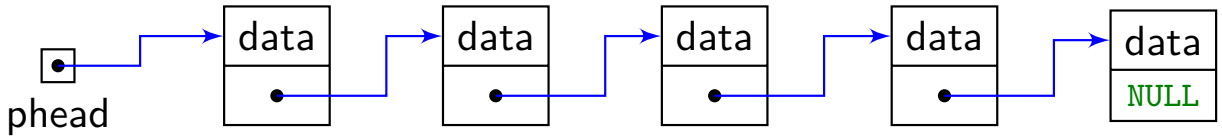


动态数组作函数参数

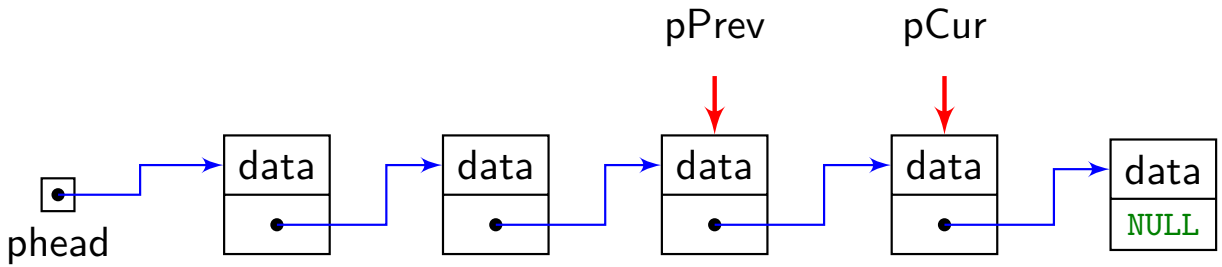
```c
#include <stdio.h>
#include <stdlib.h>
...
int main()
{
   int *pa = NULL;
   ...
   /* 分配内存 */
   pa = (int*)malloc(n * sizeof(int));
   ...
   Traversal(pa, n, Input);
   ...
   Traversal(pa, n, Output);
   ...
   // 释放内存
   free(pa);
   pa = NULL; //防止悬空指针
   return 0;
}
void Traversal(int *pa, int n, void (*pf)(int *))
{
   int *p;
   for(p = pa; p < pa + n; p++){
      pf(p); // 调用传入的函数
   }
}
void Input(int * pData)
{
   scanf("%d", pData);
}
void Output(int * pData)
{
   printf("%d ", *pData);
}
```



为链表添加节点 (无头节点)

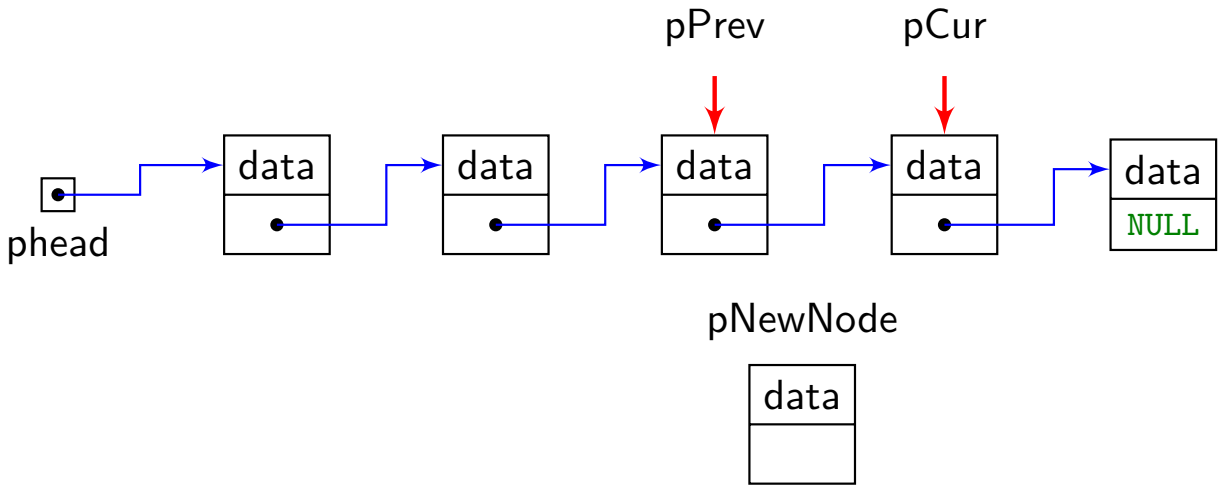


为链表添加节点 (无头节点)



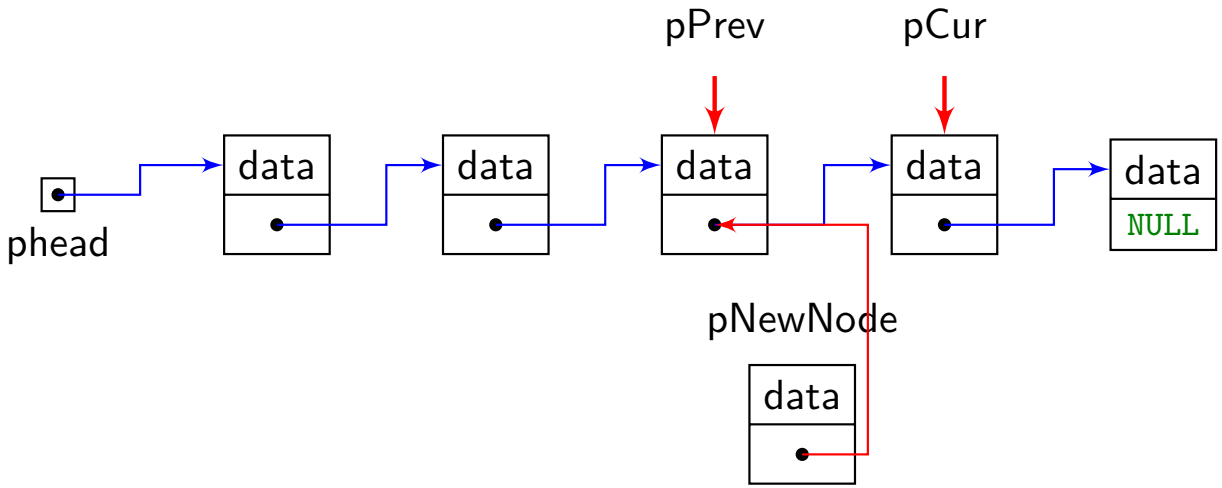

```
// 功能：根据关键字得到当前节点的指针和前驱节点的指针
void GetPrevCurByKey(ListNode *pList, int nKey,
                     ListNode **pPrev, ListNode **pCur)
{
    // p 记录当前节点指针，q 记录前驱节点的指针
    ListNode *p = pList, *q = NULL;

    while(p != NULL && p->data != nKey)
    {
        q = p; // 调整前驱节点的指针为当前节点
        p = p->next; // 指向下一个节点
    }

    *pPrev = q; // 更新传入的指针指向内存的内容
    *pCur = p; // 更新传入的指针指向内存的内容
}
```

为链表添加节点 (无头节点)



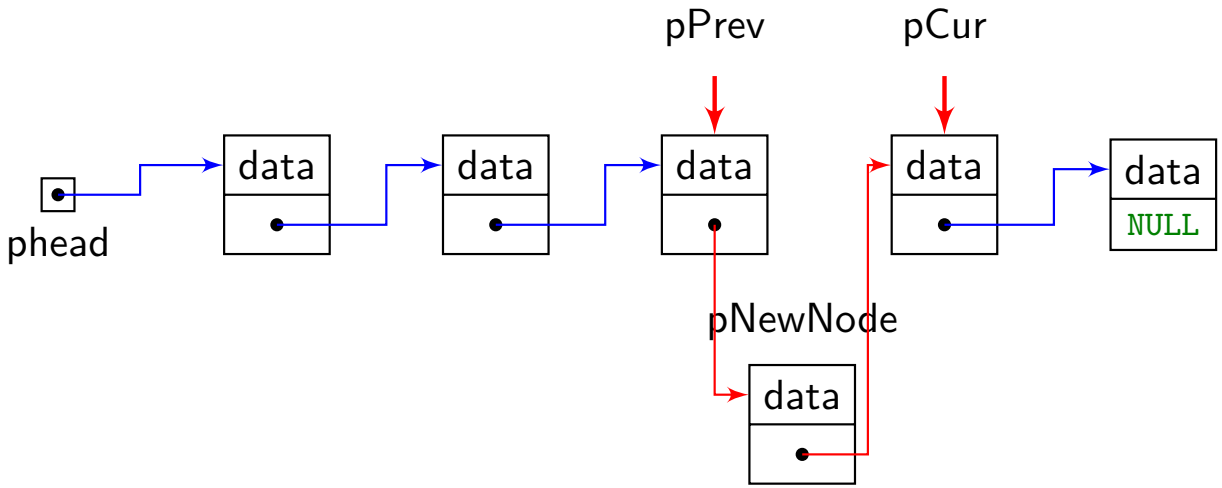

```
pNewNode = (ListNode*)malloc(sizeof(ListNode));
```

为链表添加节点 (无头节点)



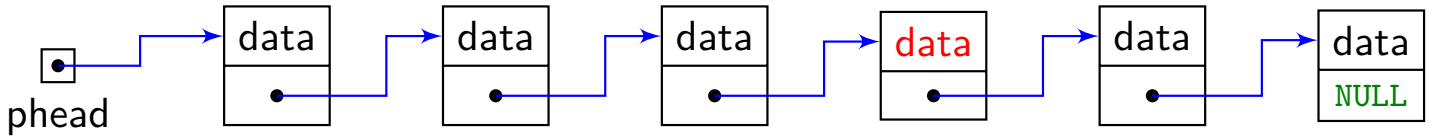

```
pNewNode->next = pPrev->next;
```

为链表添加节点 (无头节点)




```
pPrev->next = pNewNode;
```

为链表添加节点 (无头节点)




```c
// 功能：将一个节点插入到指定关键字节点之前 (无头节点)
ListNode * InsertByKeyWithoutDummyNode(ListNode *pList, int nKey, int nVal)
{
    // pCur 记录当前节点指针，pPrev 记录前驱节点的指针
    ListNode *pCur = NULL, *pPrev = NULL;
    // 获得插入点当前节点指针和前驱节点的指针
    GetPrevCurByKey(pList, nKey, &pPrev, &pCur);
    if(pCur == NULL){  // 没有找到
        return pList;
    }

    if(pPrev == NULL){ // 第 1 个节点
        pList = AddHeadRetHead(pList, nVal); // 头插法插入一个节点
    }
    else{ // 其它节点
        ListNode *pNewNode = (ListNode *)malloc(sizeof(ListNode));
        if(pNewNode == NULL){
            return pList; // 创建新节点失败，返回原头指针
        }
        pNewNode->data = nVal;
        pNewNode->next = pPrev->next;
        pPrev->next = pNewNode;
    }
    return pList;
}
```