TP3 Pointeurs
(Solution Ex 3.3 Séparation Négatifs/Positifs)

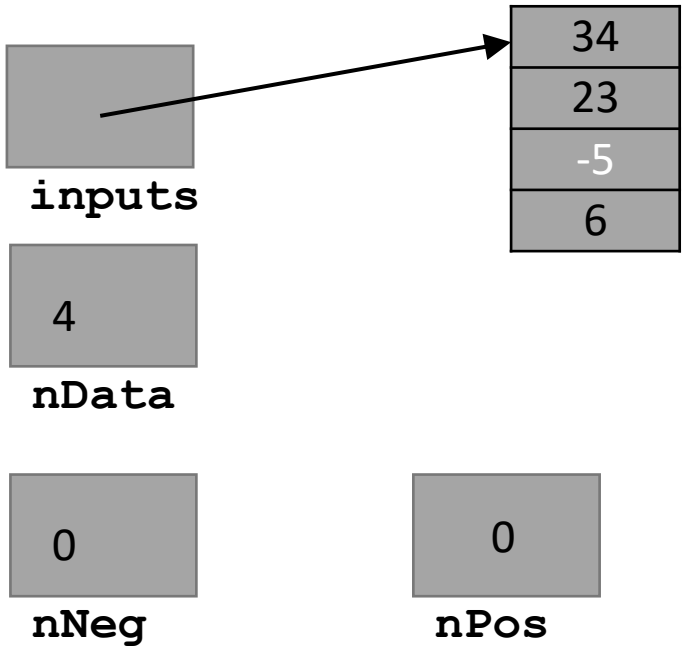
Partie 2 – Solution avec pointeurs

Phase 1

Remplissage tableau et comptage

Remplissage tableau et comptage

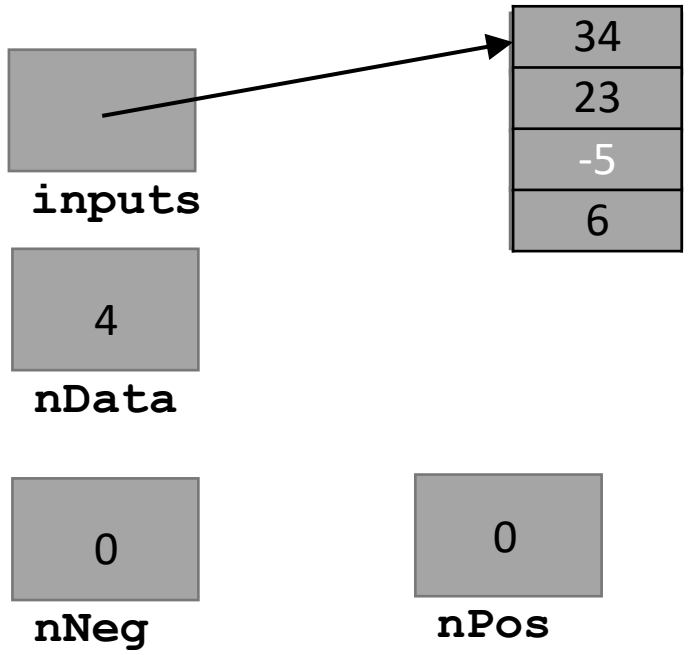
```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", &(inputs[i]));
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```



Remplissage tableau et comptage

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", &(inputs[i]));
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```

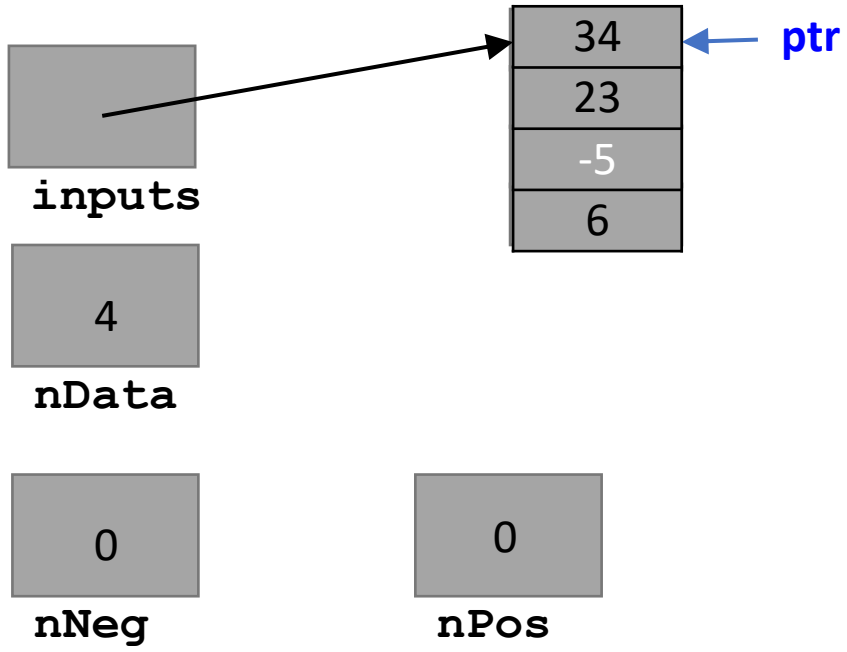
```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", inputs+i);
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```



Remplissage tableau et comptage

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", &(inputs[i]));
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int* ptr=inputs; ptr-inputs<nData; ptr++){
   scanf("%d", ptr);
   if (*ptr>=0)
      nPos++;
   else
      nNeg++;
}
```

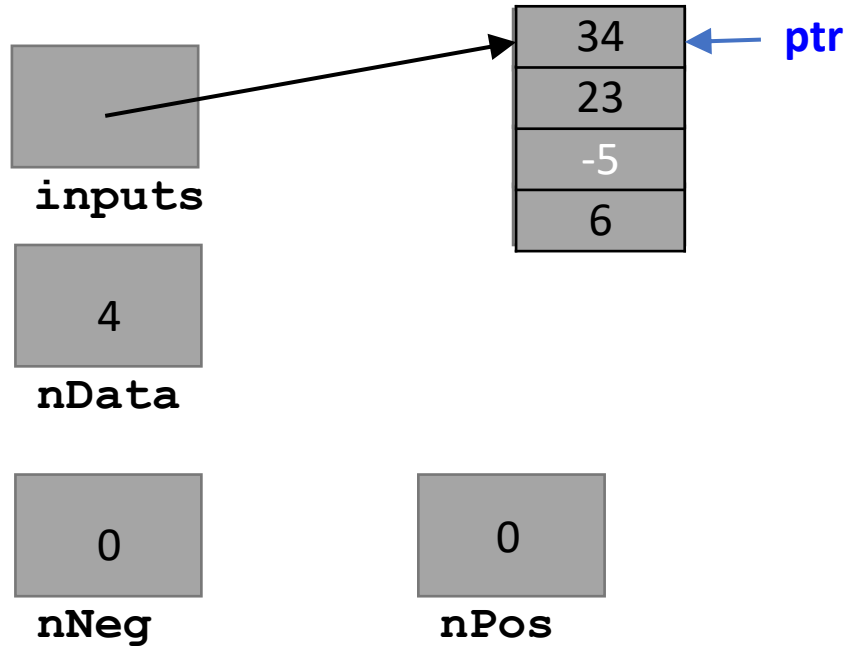


Remplissage tableau et comptage

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", &(inputs[i]));
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int* ptr=inputs; ptr-inputs<nData; ptr++){
   scanf("%d", ptr);
   if (*ptr>=0)
      nPos++;
   else
      nNeg++;
}
```

Définition d'un baladeur
de type pointeur vers `int`

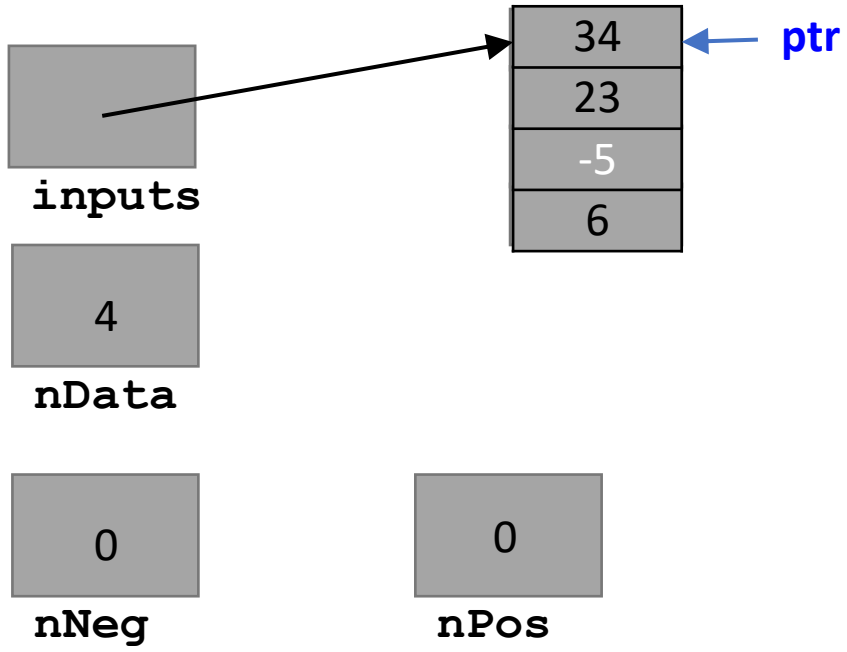


Remplissage tableau et comptage

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", &(inputs[i]));
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int* ptr=inputs; ptr-inputs<nData; ptr++){
   scanf("%d", ptr);
   if (*ptr>=0)
      nPos++;
   else
      nNeg++;
}
```

pas de & car int*

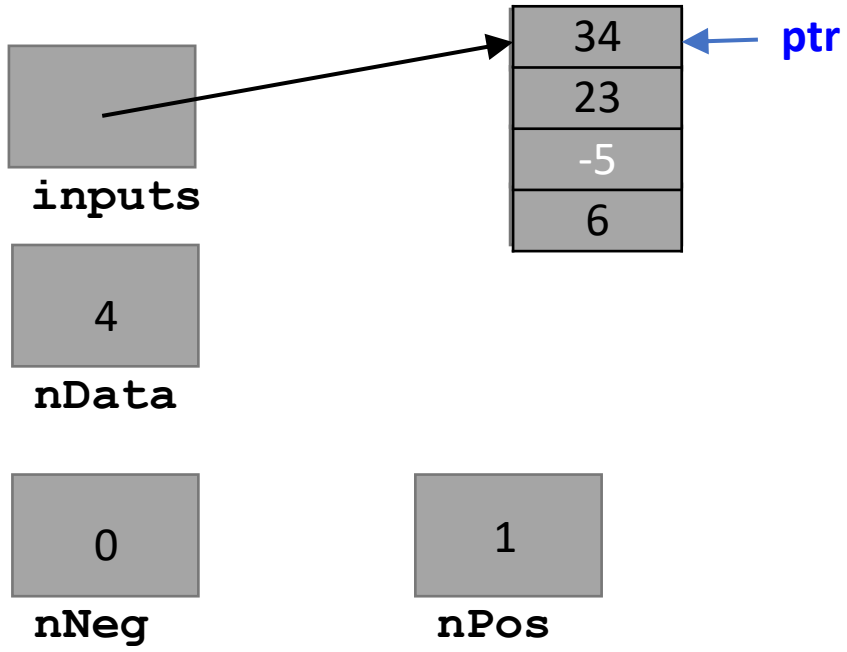


Remplissage tableau et comptage

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", &(inputs[i]));
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int* ptr=inputs; ptr-inputs<nData; ptr++){
   scanf("%d", ptr);
   if (*ptr>=0)
      nPos++;
   else
      nNeg++;
}
```

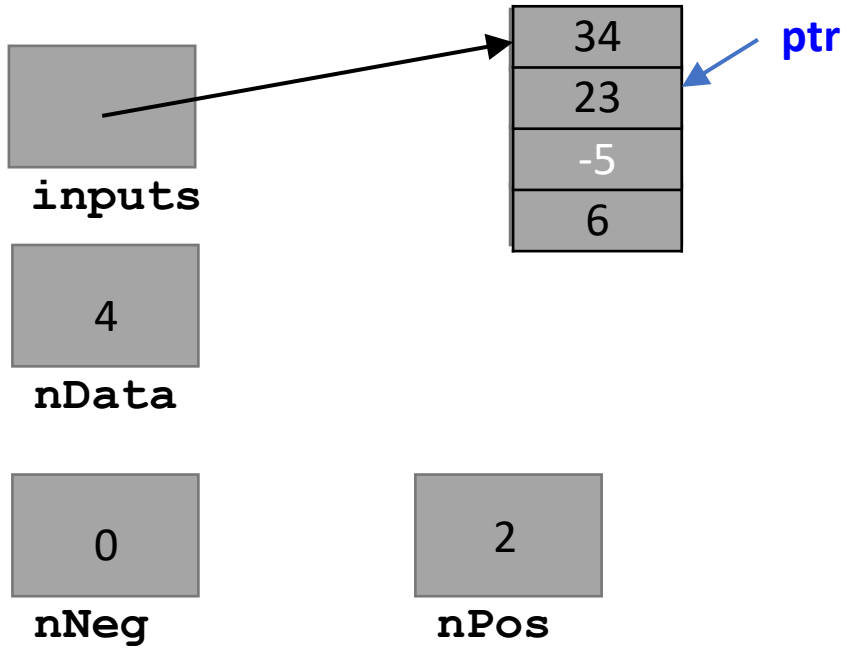
déréférencement



Remplissage tableau et comptage

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", &(inputs[i]));
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```

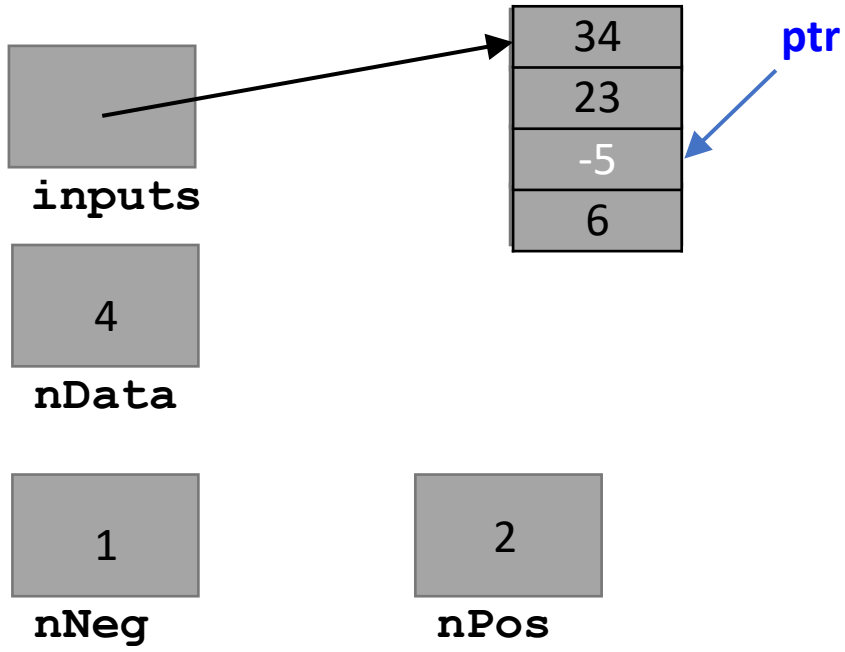
```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int* ptr=inputs; ptr-inputs<nData; ptr++){
   scanf("%d", ptr);
   if (*ptr>=0)
      nPos++;
   else
      nNeg++;
}
```



Remplissage tableau et comptage

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", &(inputs[i]));
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```

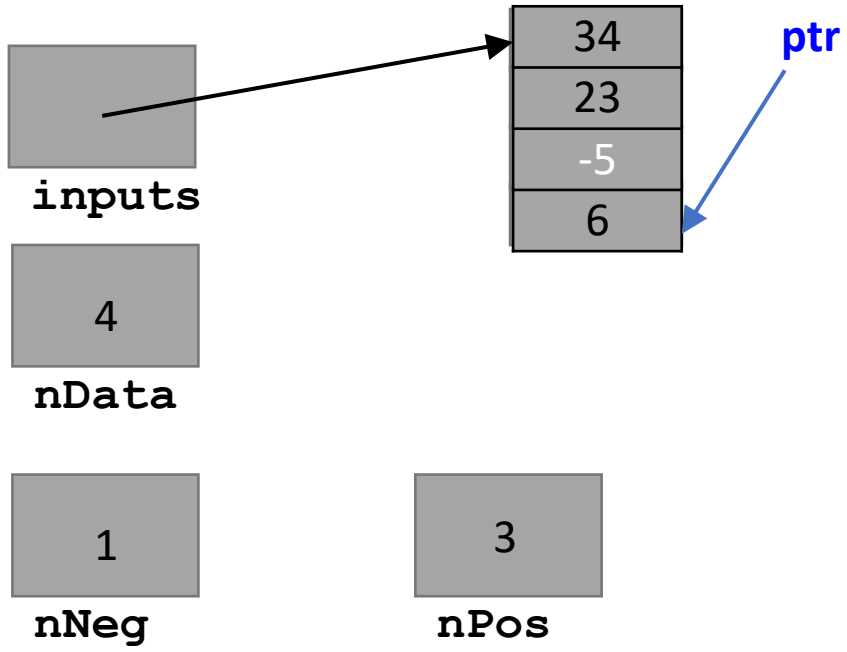
```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int* ptr=inputs; ptr-inputs<nData; ptr++){
   scanf("%d", ptr);
   if (*ptr>=0)
      nPos++;
   else
      nNeg++;
}
```



Remplissage tableau et comptage

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", &(inputs[i]));
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```

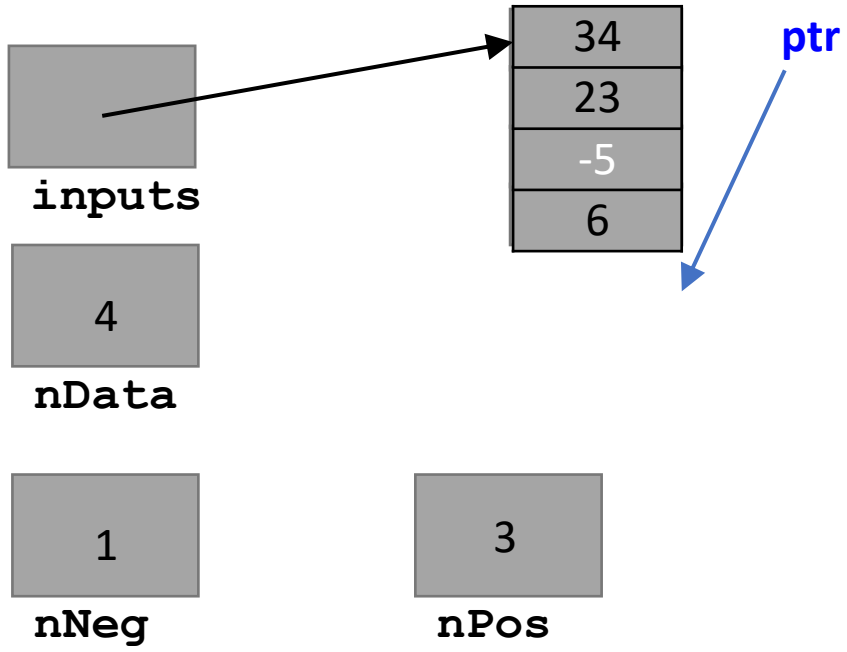
```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int* ptr=inputs; ptr-inputs<nData; ptr++){
   scanf("%d", ptr);
   if (*ptr>=0)
      nPos++;
   else
      nNeg++;
}
```



Remplissage tableau et comptage

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int i=0; i<nData; i++) {
   scanf("%d", &(inputs[i]));
   if (inputs[i]>=0)
      nPos++;
   else
      nNeg++;
}
```

```
// Read inputs and counts neg and pos
int nPos = nNeg = 0;
printf("Entrez les donnees:\n");
for (int* ptr=inputs; ptr-inputs<nData; ptr++){
   scanf("%d", ptr);
   if (*ptr>=0)
      nPos++;
   else
      nNeg++;
}
```

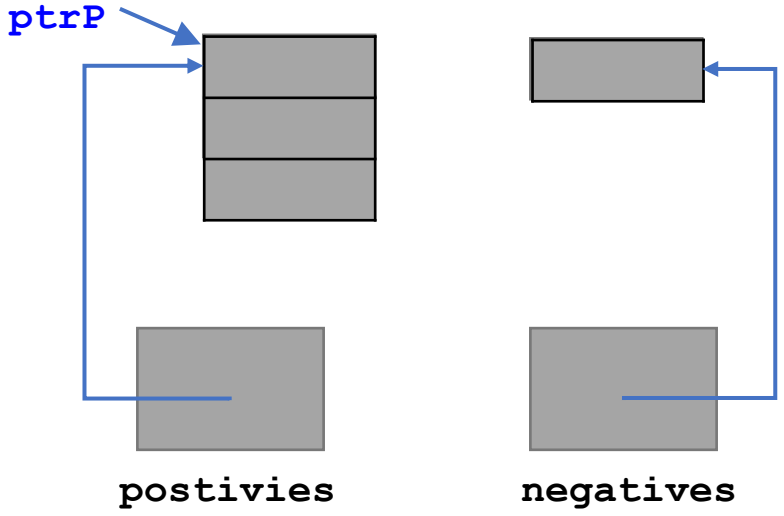


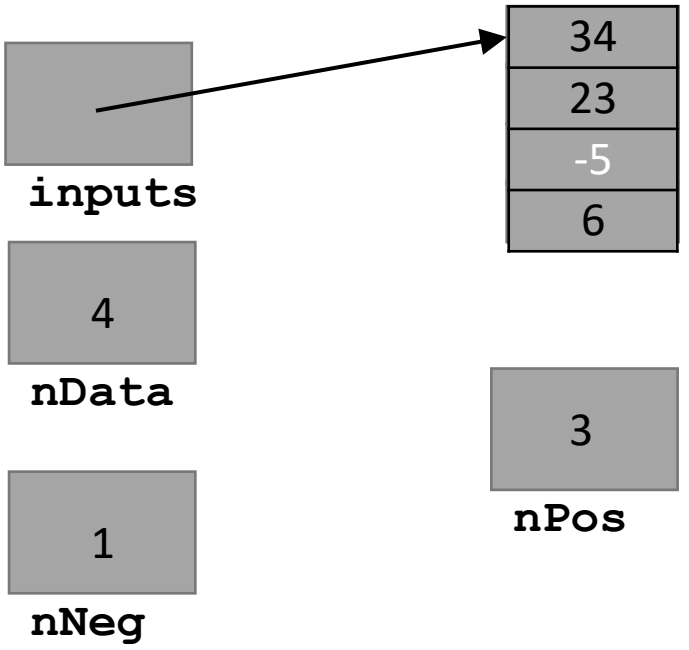
Phase 2

Remplissage tableaux négatifs et positifs

Remplissage tableaux négatifs et positifs

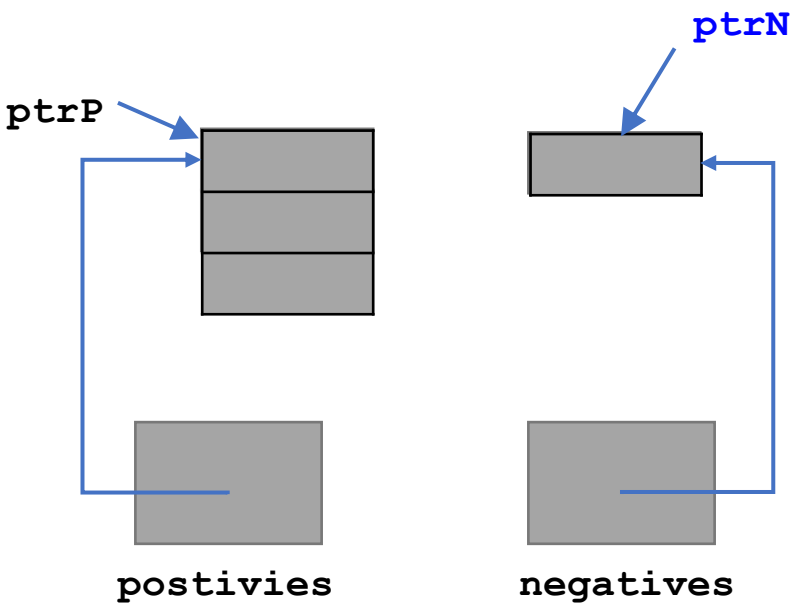
```
// Fill pos and neg arrays
int* ptrP = positives;
int* ptrN = negatives;
for (int* ptr = inputs; ptr-inputs<nData; ptr++) {
   if (*ptr>=0) {
      *ptrP = *ptr;
      ptrP++;
   }
   else {
      *ptrN = *ptr;
      ptrN++;
   }
}
```

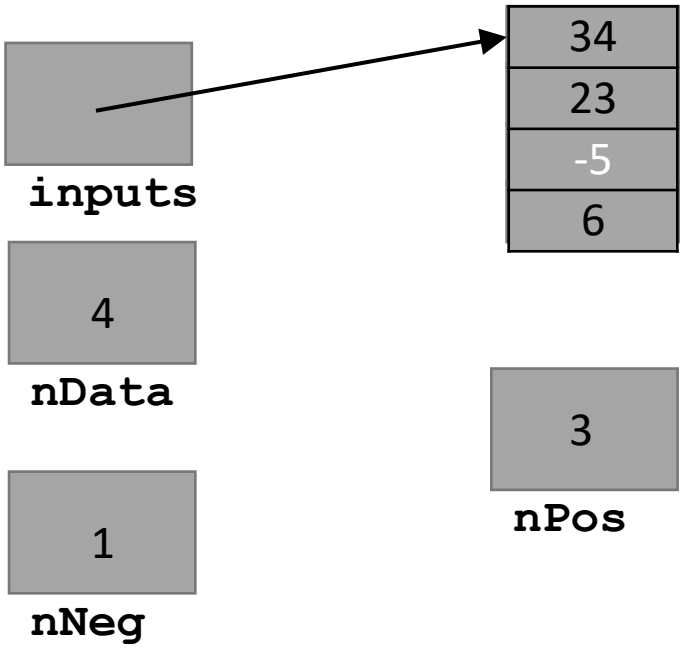




Remplissage tableaux négatifs et positifs

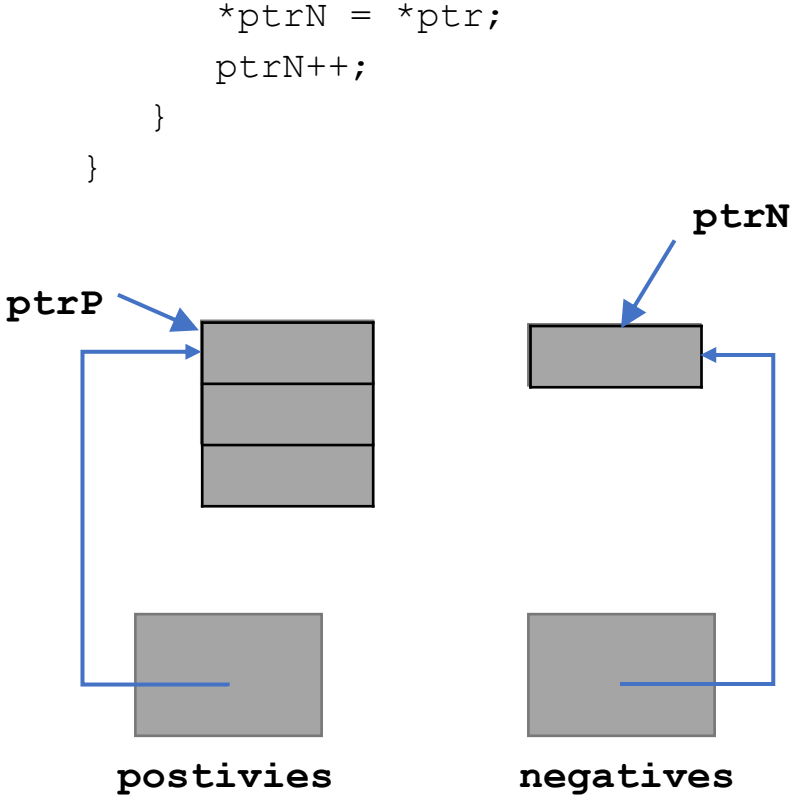
```
// Fill pos and neg arrays
int* ptrP = positives;
int* ptrN = negatives;
for (int* ptr = inputs; ptr-inputs<nData; ptr++) {
   if (*ptr>=0) {
      *ptrP = *ptr;
      ptrP++;
   }
   else {
      *ptrN = *ptr;
      ptrN++;
   }
}
```

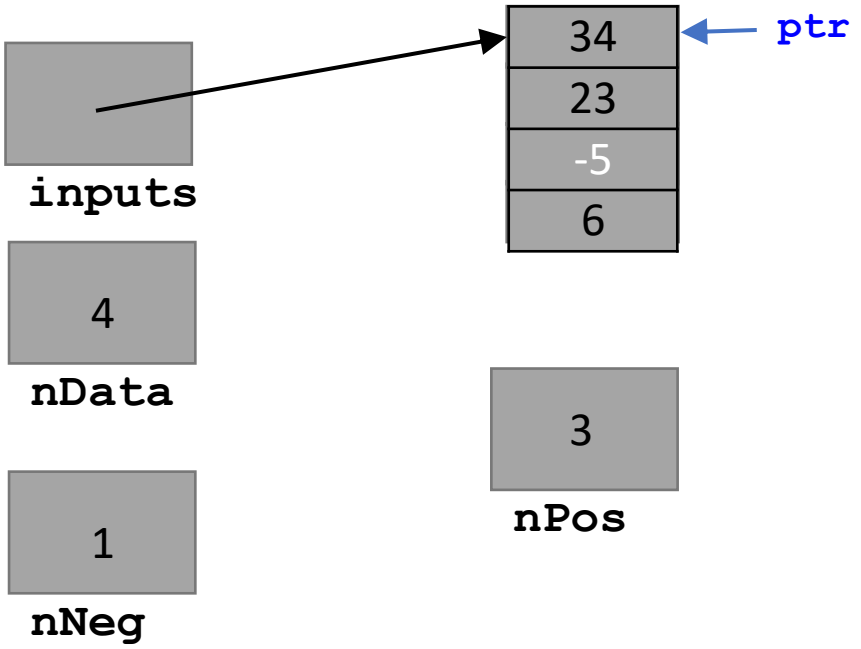




Remplissage tableaux négatifs et positifs

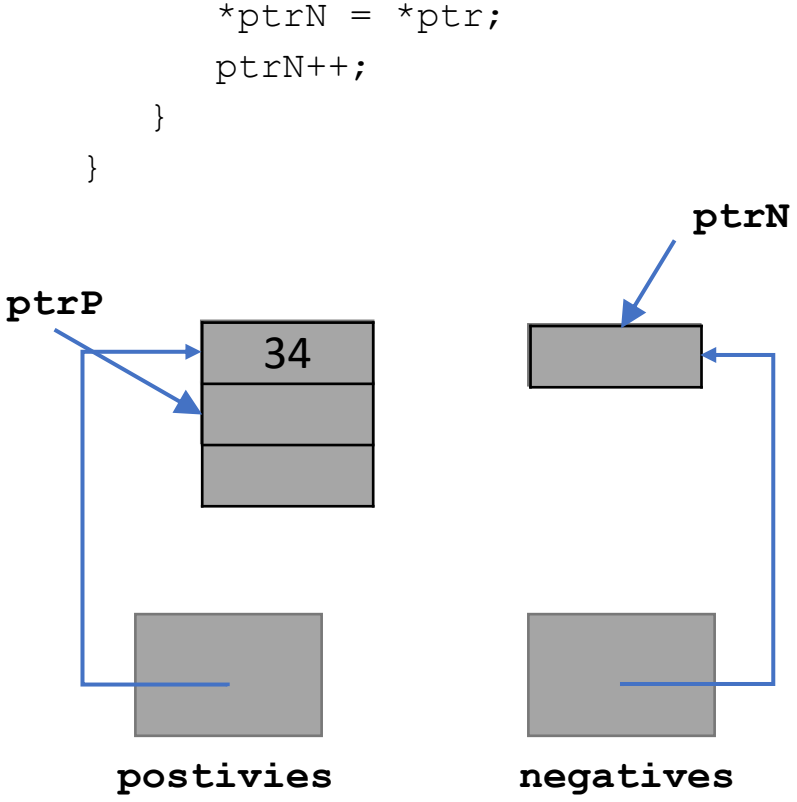
```
// Fill pos and neg arrays
int* ptrP = positives;
int* ptrN = negatives;
for (int* ptr = inputs; ptr-inputs<nData; ptr++) {
   if (*ptr>=0) {
      *ptrP = *ptr;
      ptrP++;
   }
   else {
      *ptrN = *ptr;
      ptrN++;
   }
}
```

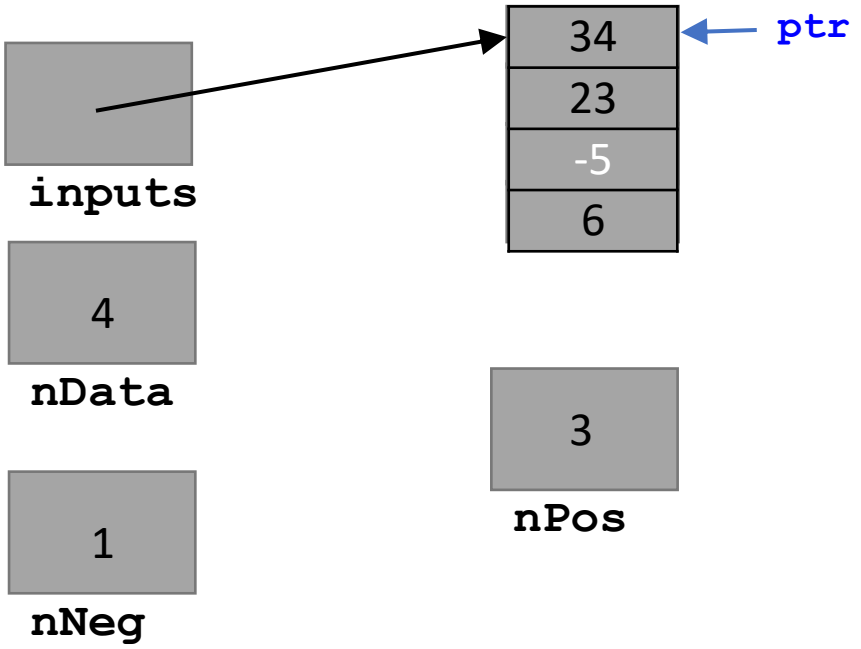




Remplissage tableaux négatifs et positifs

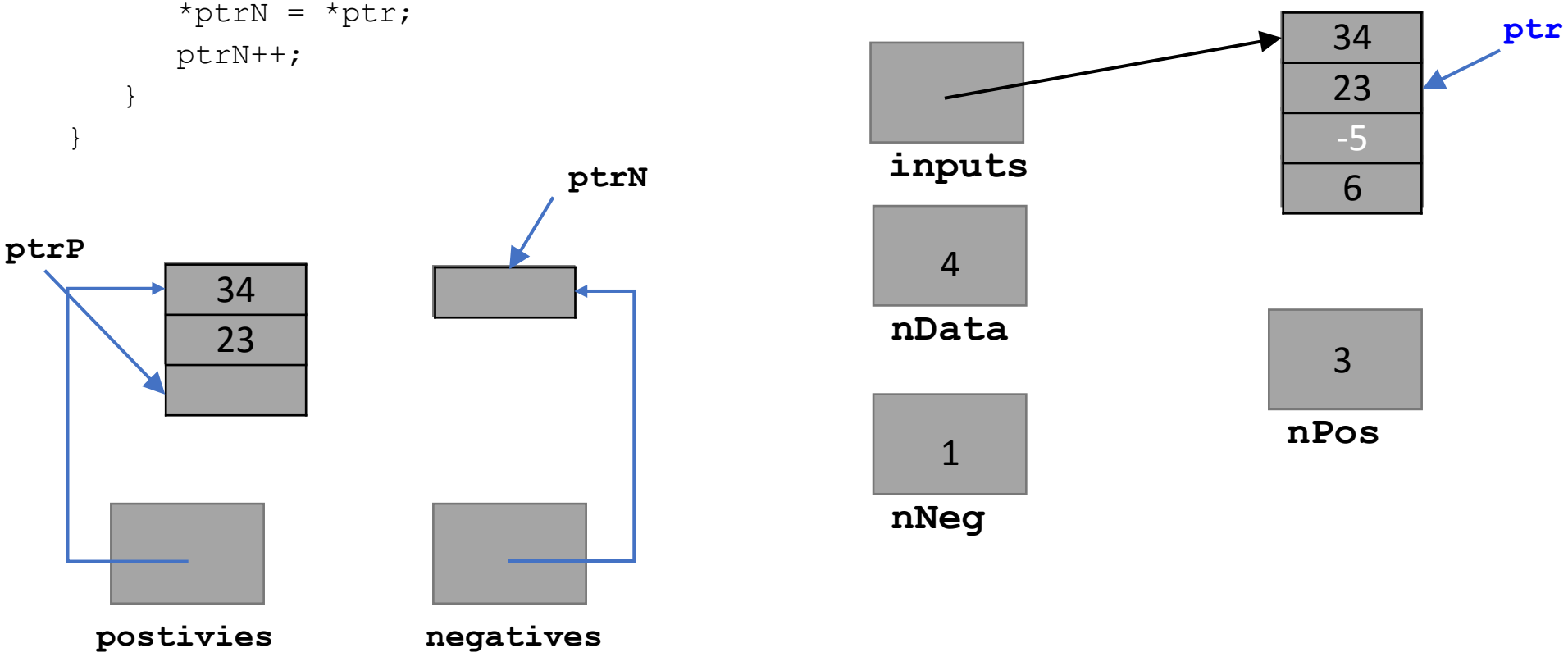
```
// Fill pos and neg arrays
int* ptrP = positives;
int* ptrN = negatives;
for (int* ptr = inputs; ptr-inputs<nData; ptr++) {
   if (*ptr>=0) {
      *ptrP = *ptr;
      ptrP++;
   }
   else {
      *ptrN = *ptr;
      ptrN++;
   }
}
```





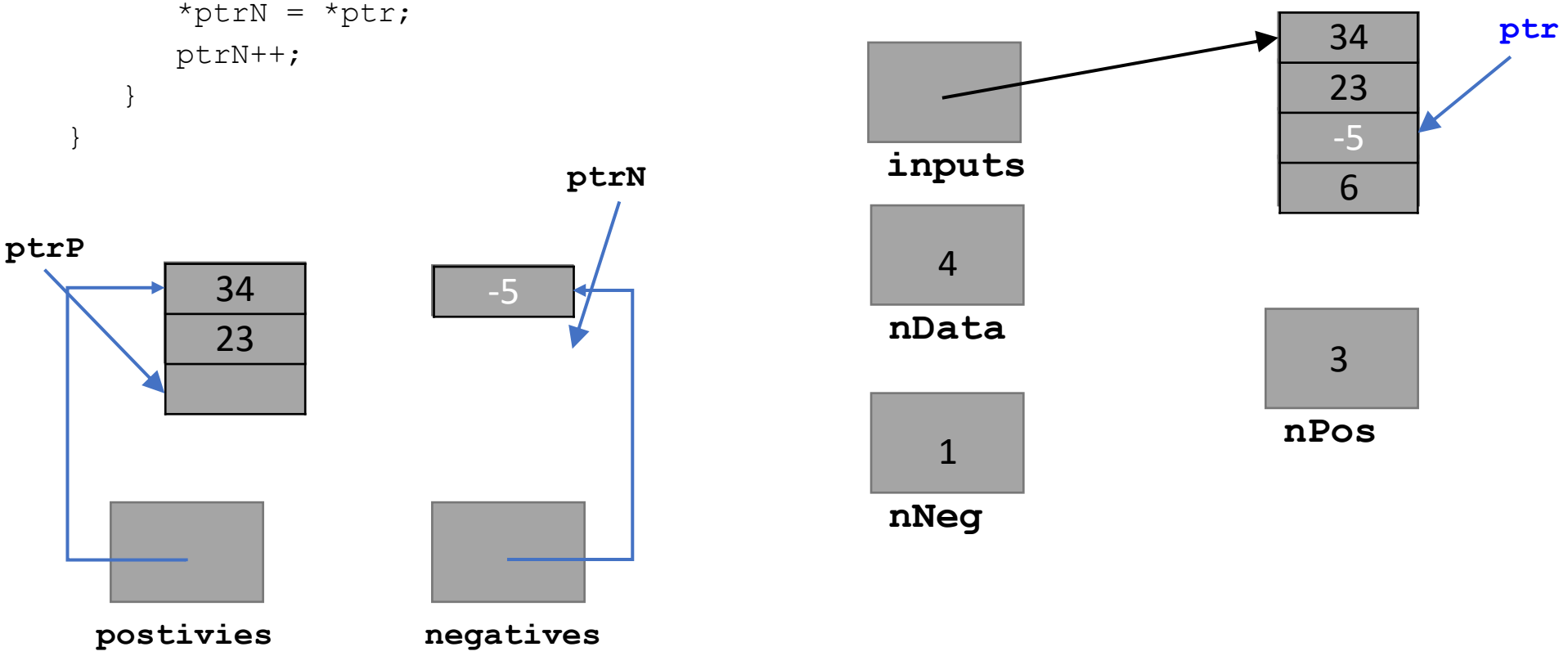
Remplissage tableaux négatifs et positifs

```
// Fill pos and neg arrays
int* ptrP = positives;
int* ptrN = negatives;
for (int* ptr = inputs; ptr-inputs<nData; ptr++) {
   if (*ptr>=0) {
      *ptrP = *ptr;
      ptrP++;
   }
   else {
      *ptrN = *ptr;
      ptrN++;
   }
}
```



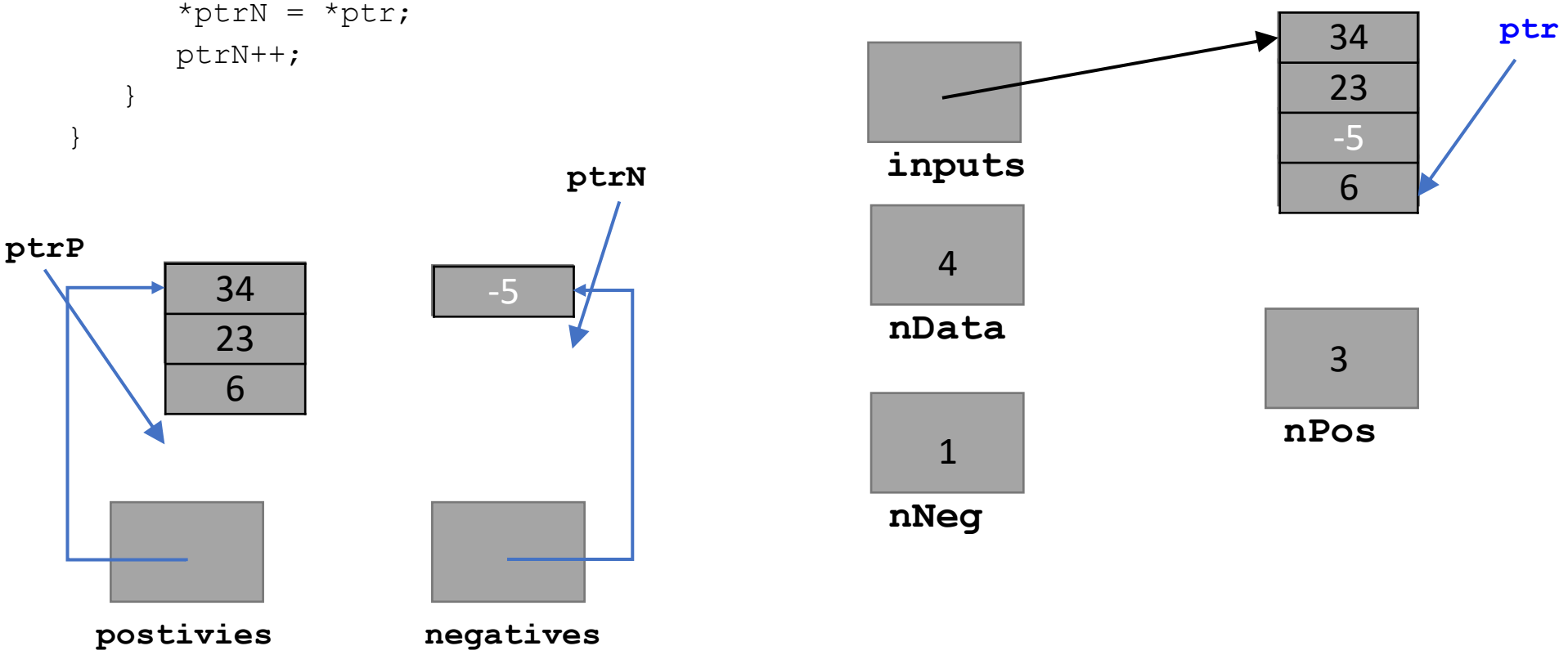
Remplissage tableaux négatifs et positifs

```
// Fill pos and neg arrays
int* ptrP = positives;
int* ptrN = negatives;
for (int* ptr = inputs; ptr-inputs<nData; ptr++) {
   if (*ptr>=0) {
      *ptrP = *ptr;
      ptrP++;
   }
   else {
      *ptrN = *ptr;
      ptrN++;
   }
}
```



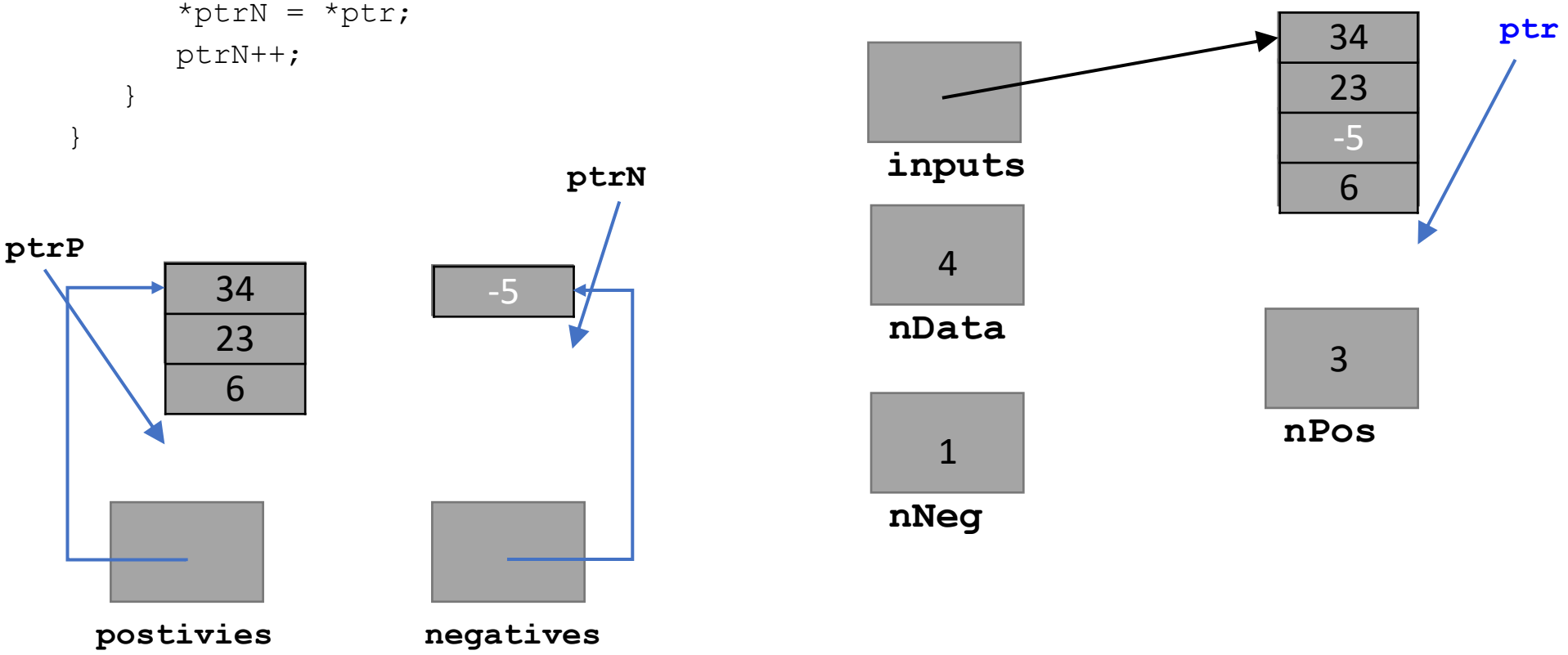
Remplissage tableaux négatifs et positifs

```
// Fill pos and neg arrays
int* ptrP = positives;
int* ptrN = negatives;
for (int* ptr = inputs; ptr-inputs<nData; ptr++) {
   if (*ptr>=0) {
      *ptrP = *ptr;
      ptrP++;
   }
   else {
      *ptrN = *ptr;
      ptrN++;
   }
}
```



Remplissage tableaux négatifs et positifs

```
// Fill pos and neg arrays
int* ptrP = positives;
int* ptrN = negatives;
for (int* ptr = inputs; ptr-inputs<nData; ptr++) {
   if (*ptr>=0) {
      *ptrP = *ptr;
      ptrP++;
   }
   else {
      *ptrN = *ptr;
      ptrN++;
   }
}
```

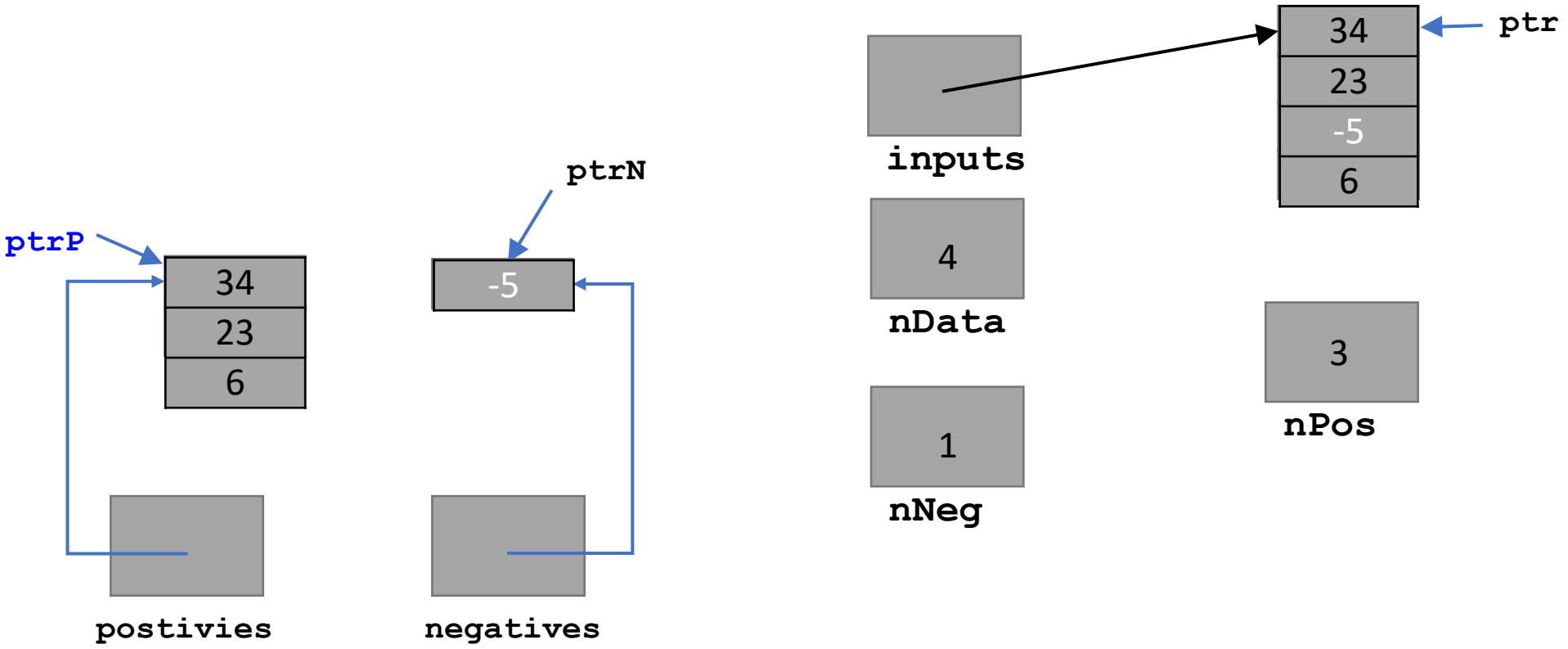


Phase 3

Affichage résultats

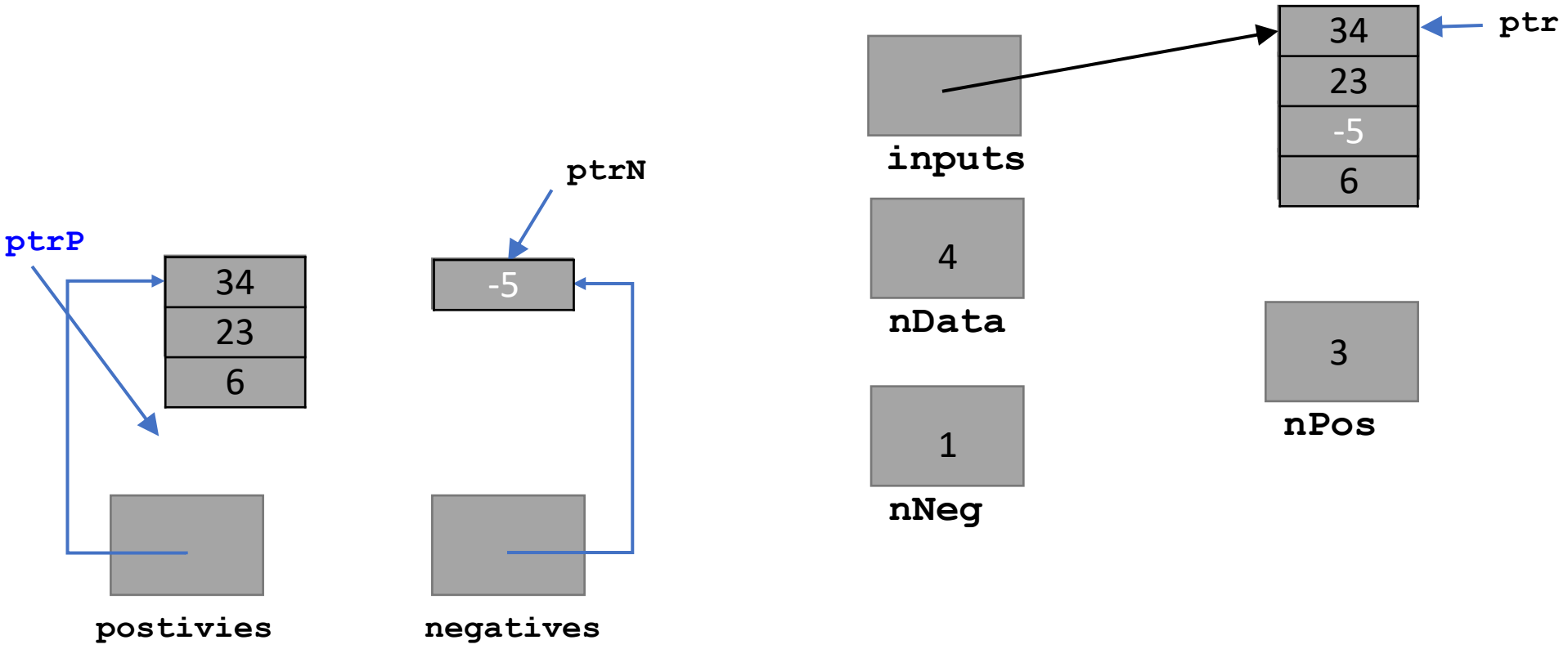
Affichage résultats

```
for (int* ptrP = positives; ptrP-positives<nPos; ptrP++)
   printf("%d ", *ptrP);
```



Affichage résultats

```
for (int* ptrP = positives; ptrP-positives<nPos; ptrP++)
   printf("%d ", *ptrP);
```



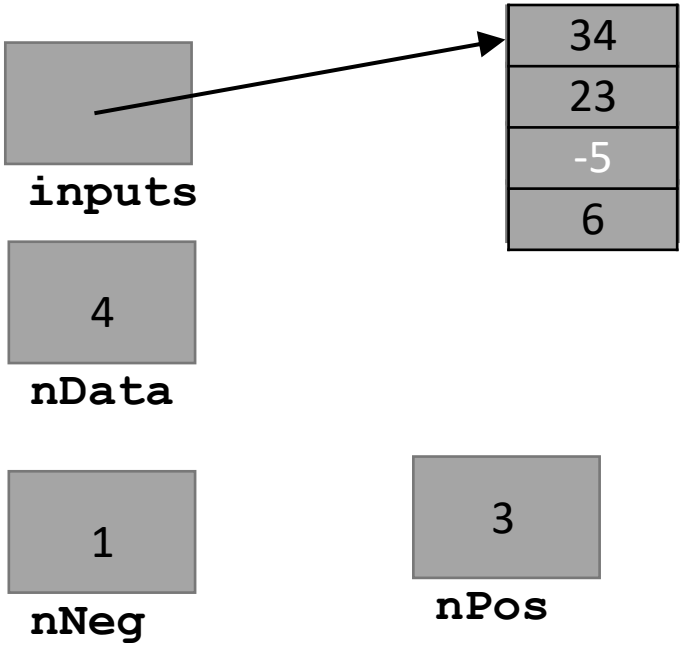
Phase 4

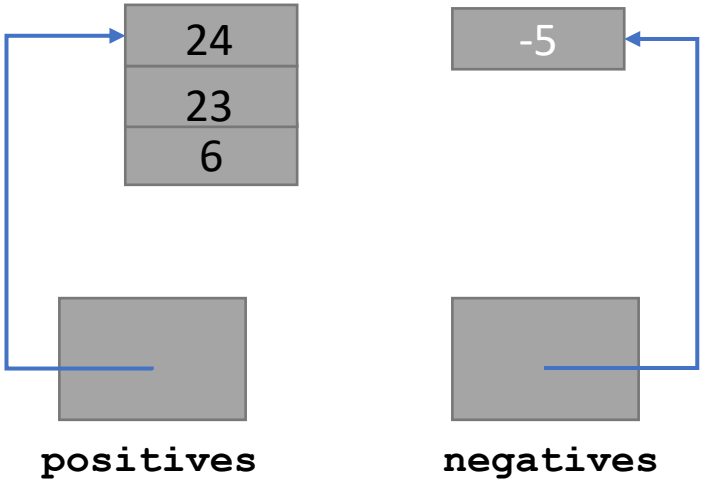
Libérations tableaux

Libérations tableaux

```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {
      printf("Entrez le nombre de donnees (0 pour terminer): ");
      scanf("%d", &nData);
      inputs = malloc(nData * sizeof(int));
      if (inputs == NULL) {
         perror("Out of memory\n");
         exit(EXIT_FAILURE);
      }
      ...
```

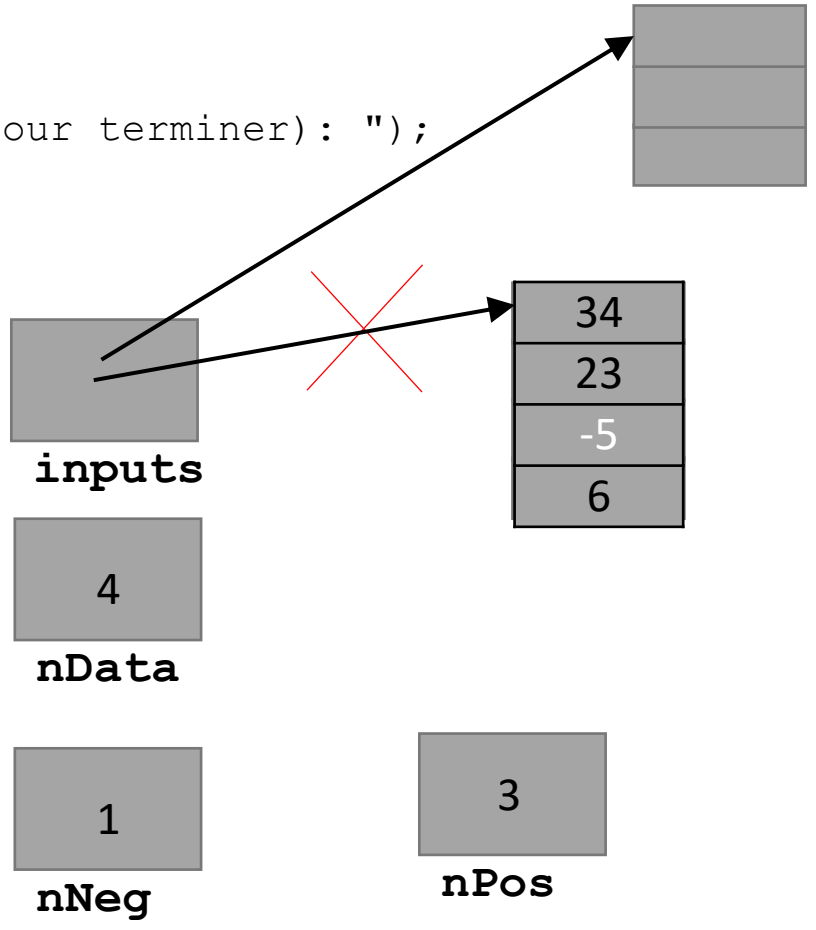
Boucle englobante :
nouveau tableau à chaque itération

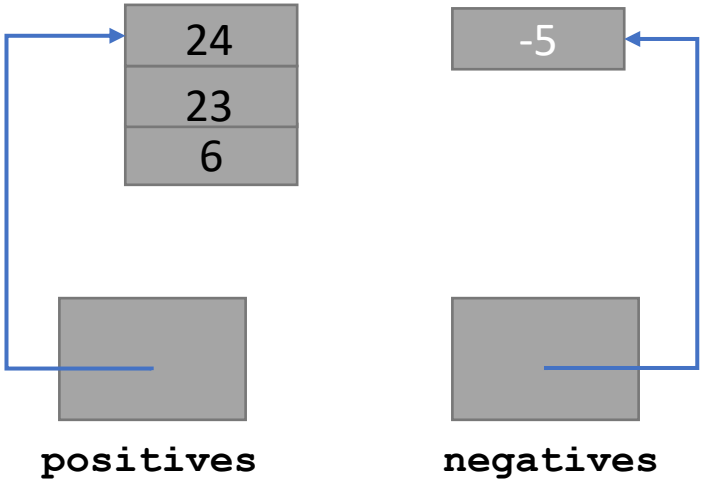




Libérations tableaux

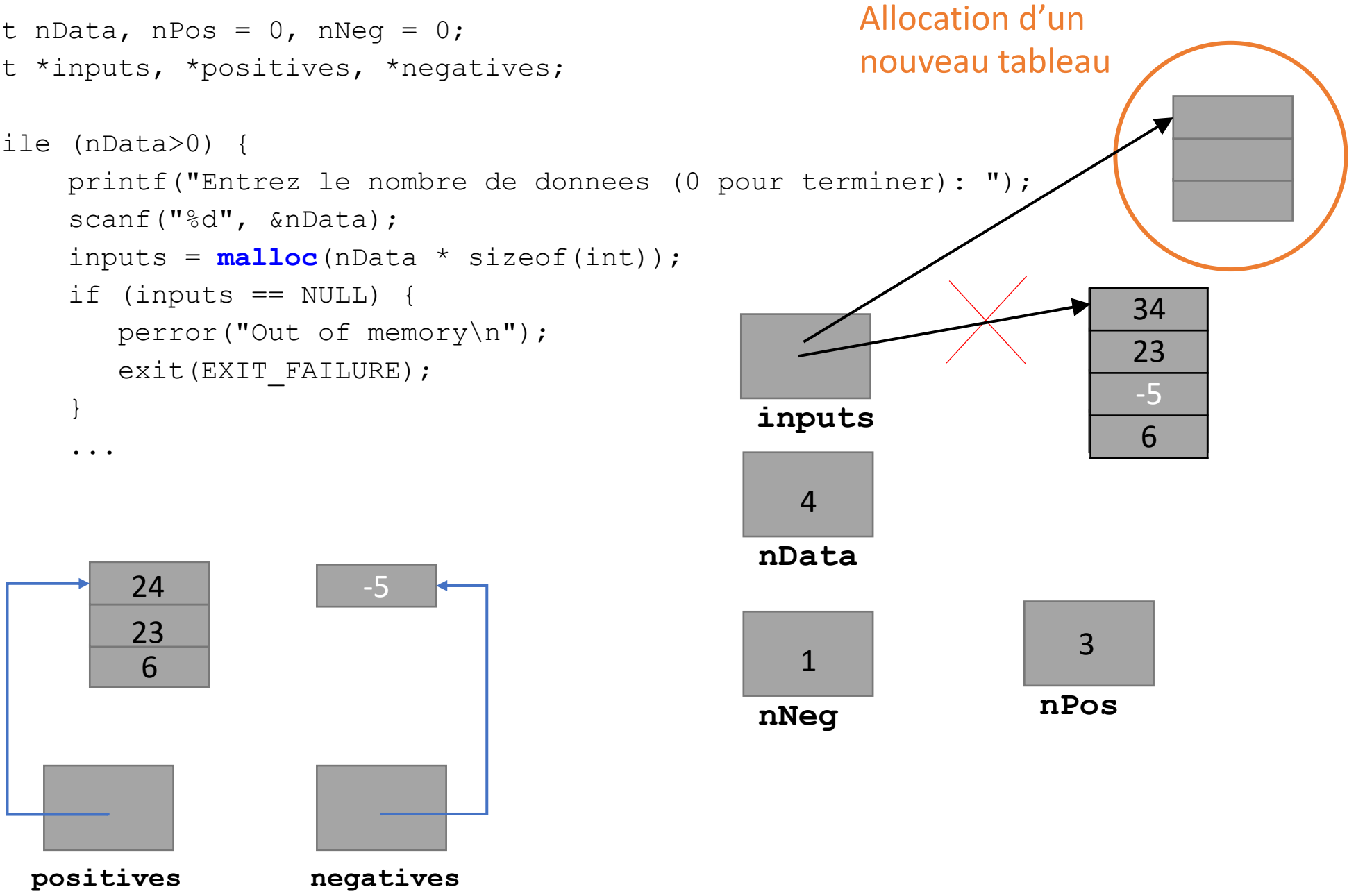
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {
      printf("Entrez le nombre de donnees (0 pour terminer): ");
      scanf("%d", &nData);
      inputs = malloc(nData * sizeof(int));
      if (inputs == NULL) {
         perror("Out of memory\n");
         exit(EXIT_FAILURE);
      }
      ...
```





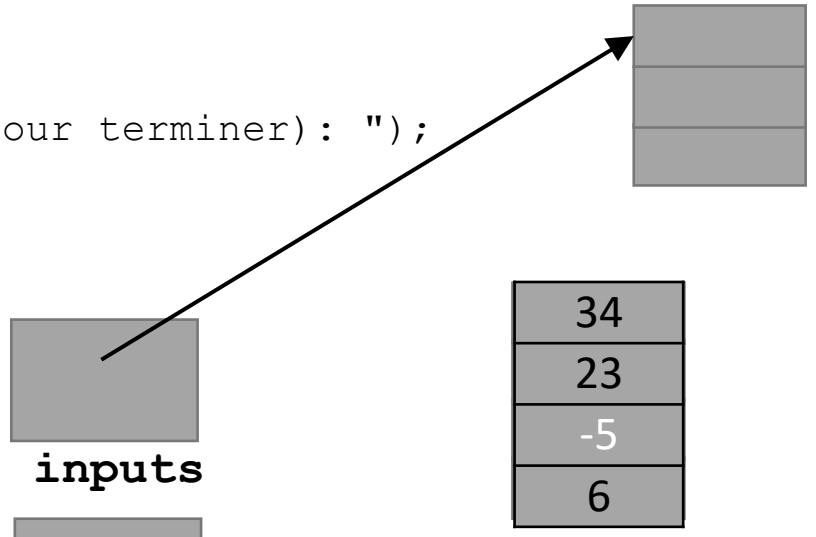
Libérations tableaux

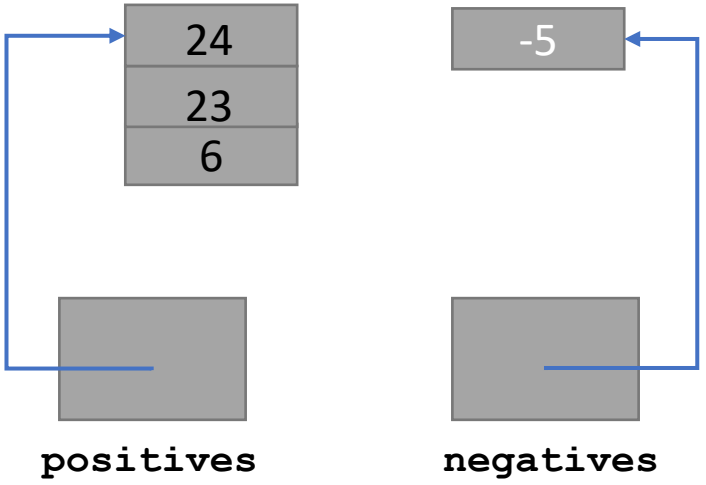
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {
      printf("Entrez le nombre de donnees (0 pour terminer): ");
      scanf("%d", &nData);
      inputs = malloc(nData * sizeof(int));
      if (inputs == NULL) {
         perror("Out of memory\n");
         exit(EXIT_FAILURE);
      }
      ...
```



Libérations tableaux

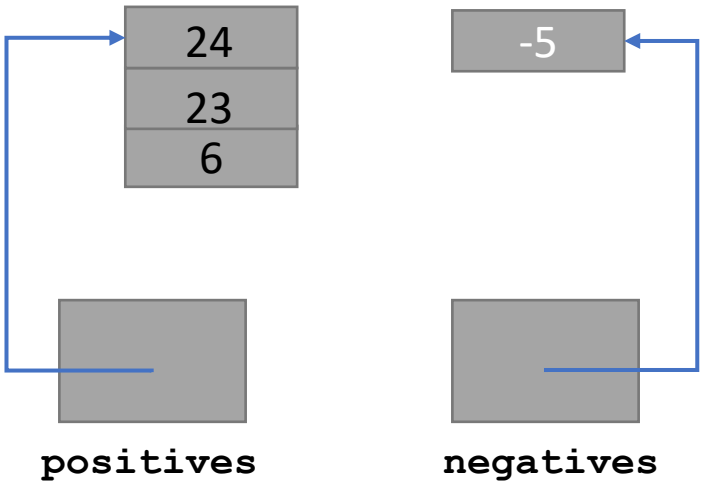
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {
      printf("Entrez le nombre de donnees (0 pour terminer): ");
      scanf("%d", &nData);
      inputs = malloc(nData * sizeof(int));
      if (inputs == NULL) {
         perror("Out of memory\n");
         exit(EXIT_FAILURE);
      }
      ...
```

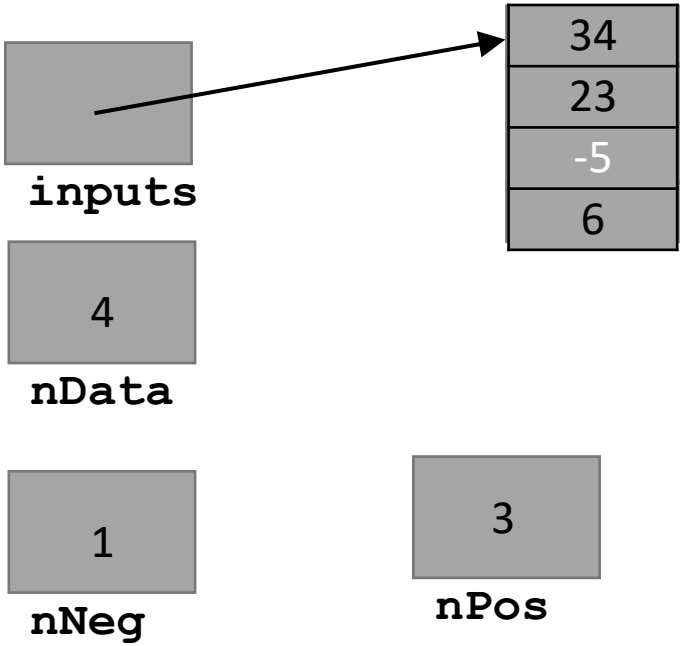




Libérations tableaux

```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {

      ...

      // Free dynamic memory
      free(inputs);
      free(negatives);
      free(positives);
}
```

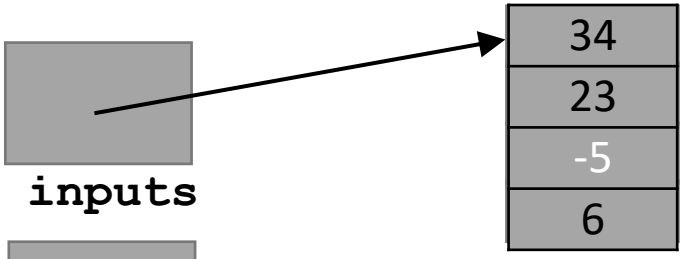


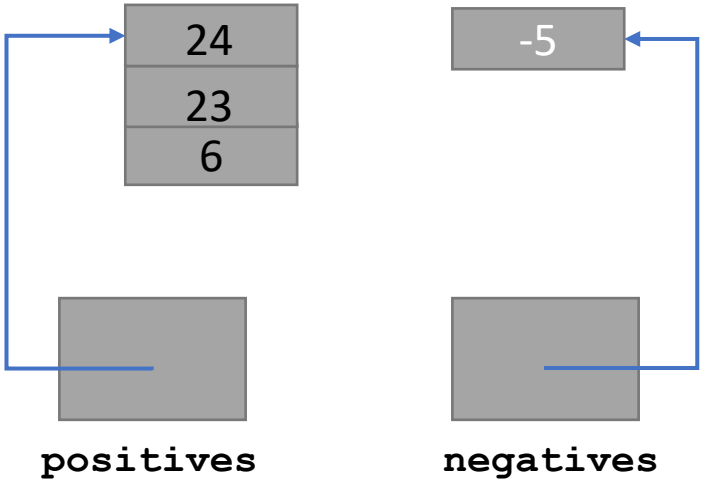


Libérations tableaux

```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {

      ...

      // Free dynamic memory
      free(inputs);
      free(negatives);
      free(positives);
}
```

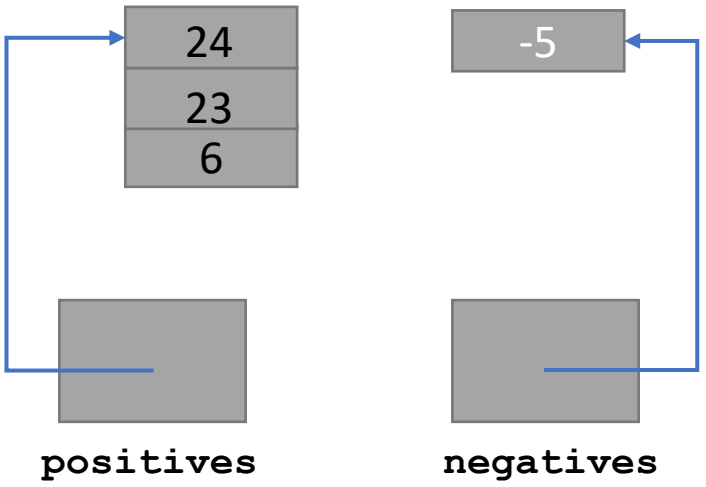
Libération des zones de mémoire dynamique

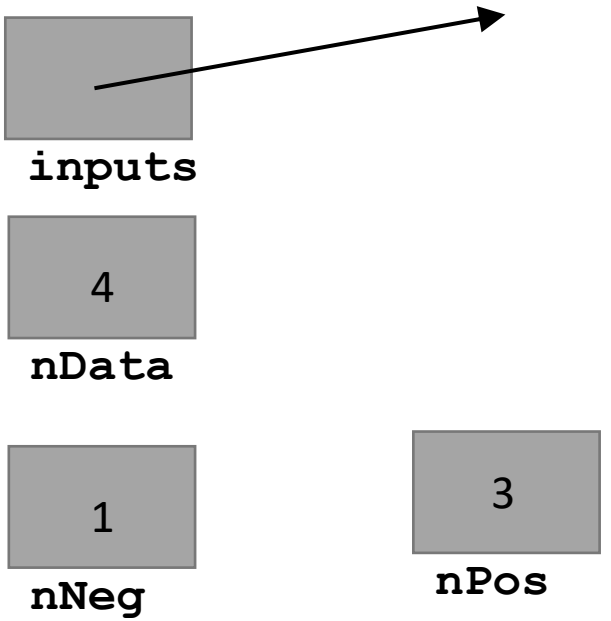




Libérations tableaux

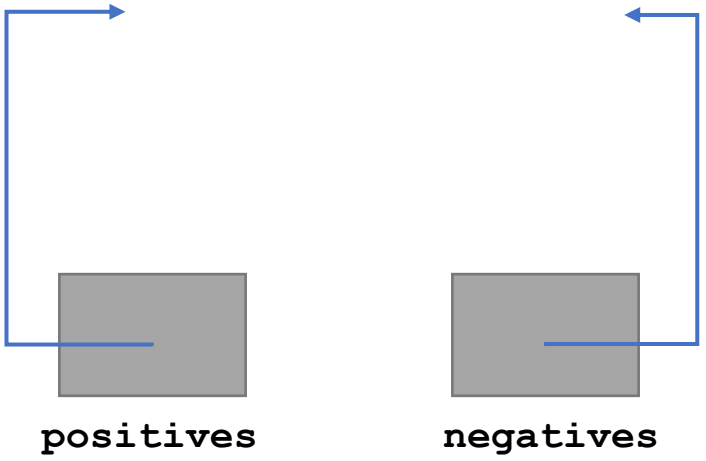
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {

      ...

      // Free dynamic memory
      free(inputs);
      free(negatives);
      free(positives);
}
```

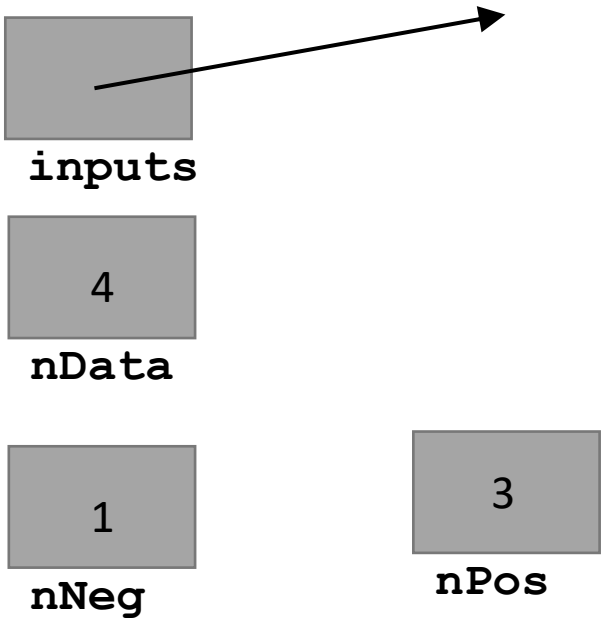




Libérations tableaux

```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {

      ...

      // Free dynamic memory
      free(inputs);
      free(negatives);
      free(positives);
}
```





Libérations tableaux

```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {

      ...

      // Free dynamic memory
      free(inputs);
      free(negatives);
      free(positives);
}
```