# TP3 Pointeurs

## (Solution Ex 3.2
## Séparation Négatifs/Positifs)

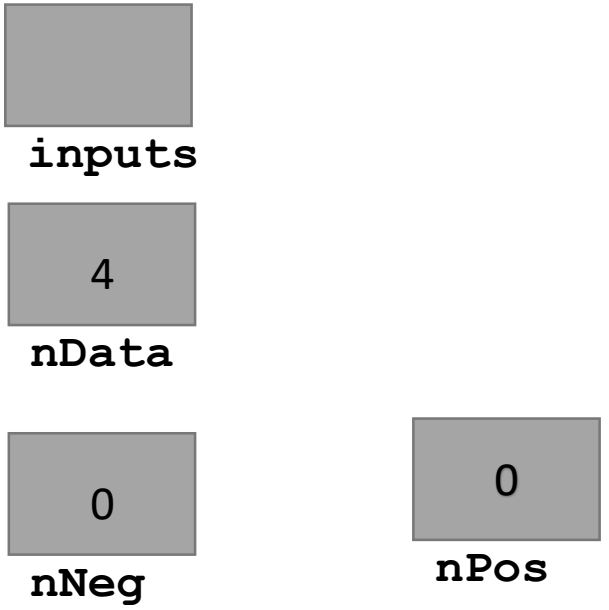
### Partie 1 – Solution avec notation indicée [ ]

# Phase 1

# Remplissage tableau et comptage

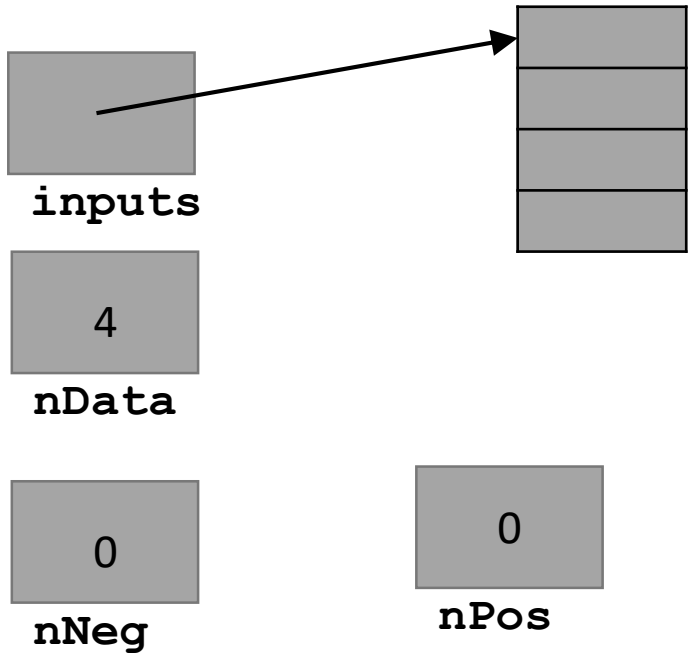
# Allocation tableau inputs

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

printf("Entrez le nombre de donnees (0 pour terminer): ");
scanf("%d", &nData);

while (nData>0) {
   inputs = malloc(nData * sizeof(int));
   if (inputs == NULL) {
      perror("Out of memory\n");
      exit(EXIT_FAILURE);
   }
   ...
```



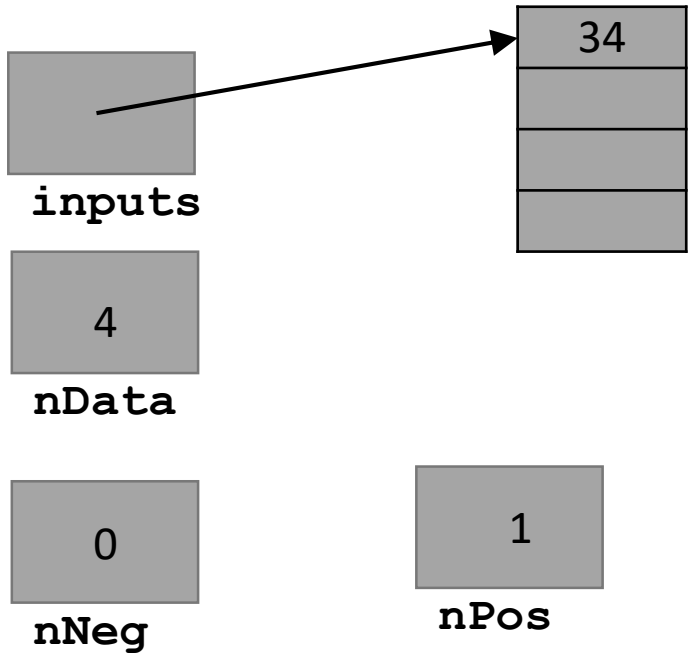
# Allocation tableau inputs

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

printf("Entrez le nombre de donnees (0 pour terminer): ");
scanf("%d", &nData);

while (nData>0) {
   inputs = malloc(nData * sizeof(int));
   if (inputs == NULL) {
      perror("Out of memory\n");
      exit(EXIT_FAILURE);
   }
   ...
```



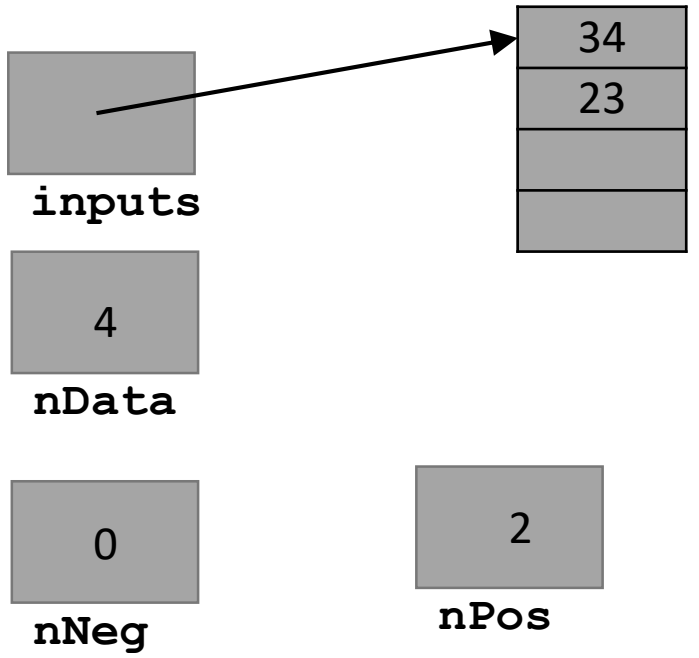
# Remplissage tableau et comptage

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

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



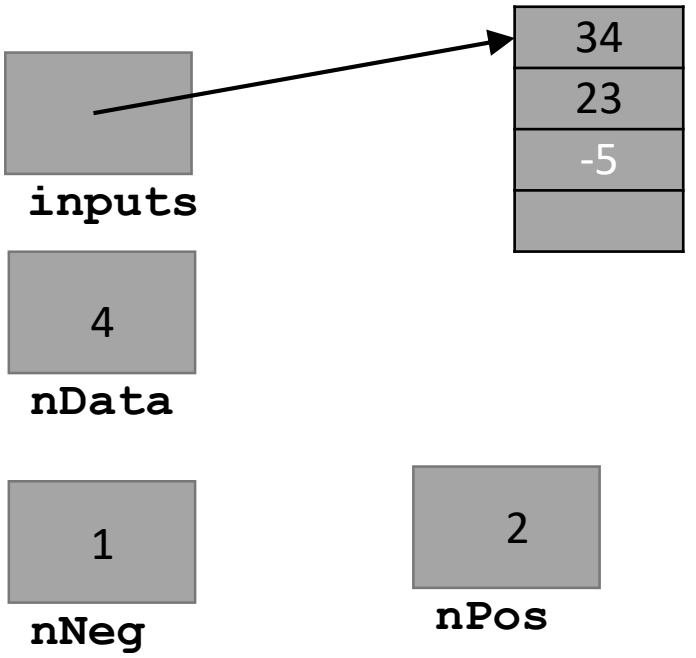
# Remplissage tableau et comptage

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

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



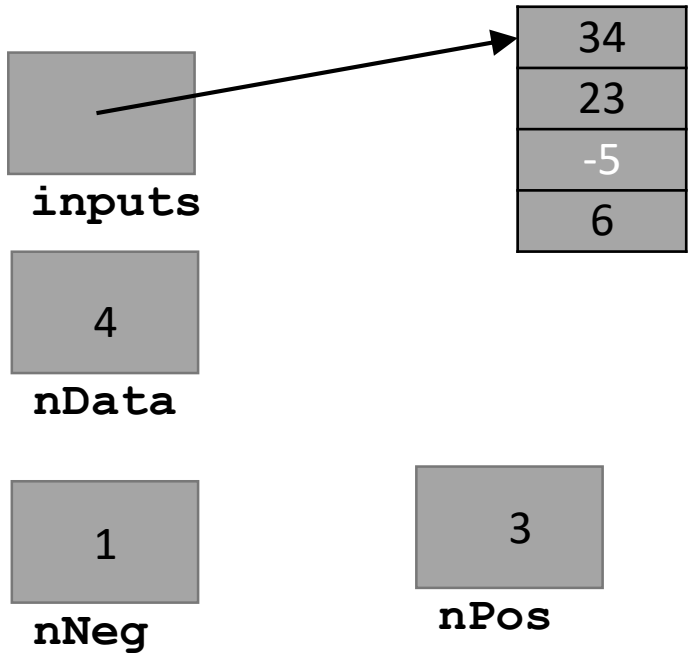
# Remplissage tableau et comptage

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

while (nData>0) {
   ...
   // Read inputs and counts neg and pos
   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

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

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

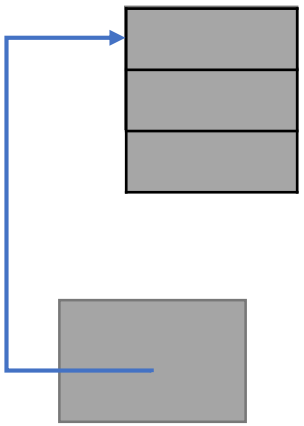


# Phase 2

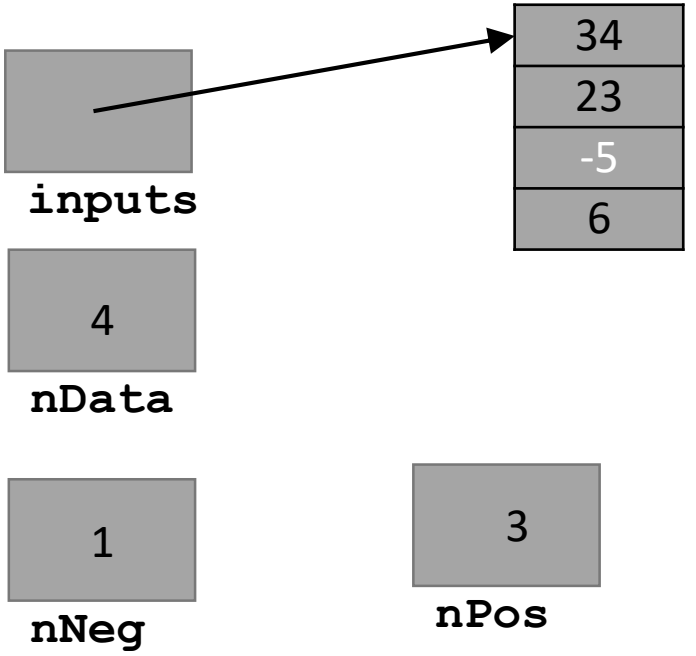
# Remplissage tableaux négatifs et positifs

# Allocation tableaux pos et neg

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

while (nData>0) {
   ...
   positives = malloc(nPos * sizeof(int));
   if (positives == NULL) {
      perror("Out of memory\n");
      exit(EXIT_FAILURE);
   }
   ...
```

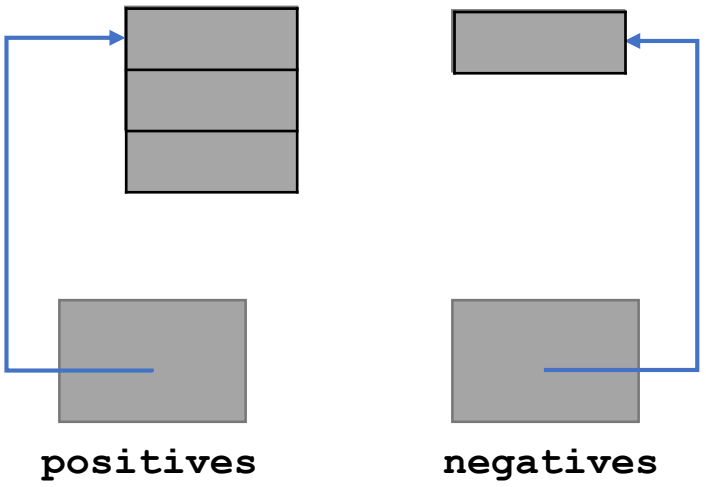


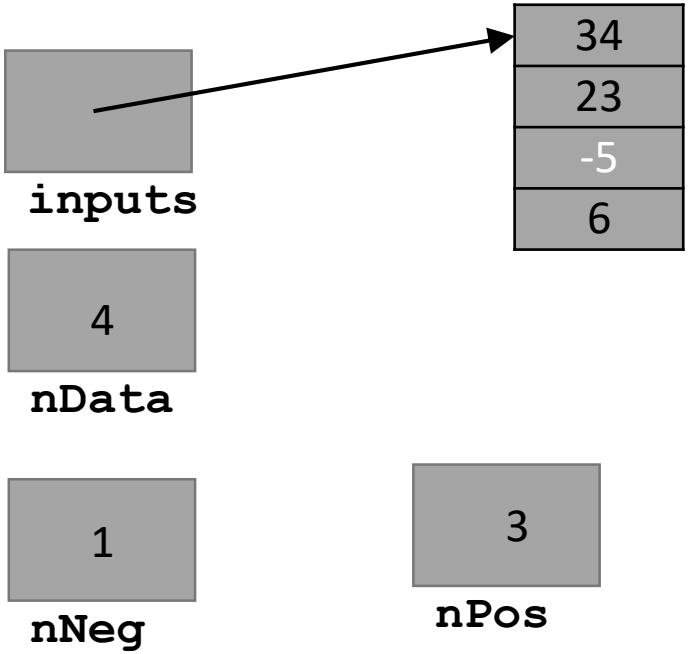
**positives**



# Allocation tableaux pos et neg

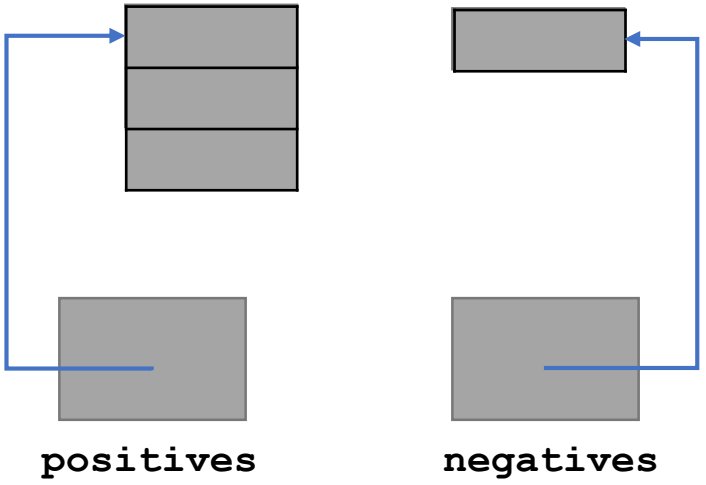
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {
   ...
   negatives = malloc(nNeg * sizeof(int));
   if (negatives == NULL) {
      perror("Out of memory\n");
      exit(EXIT_FAILURE);
   }
   ...
```

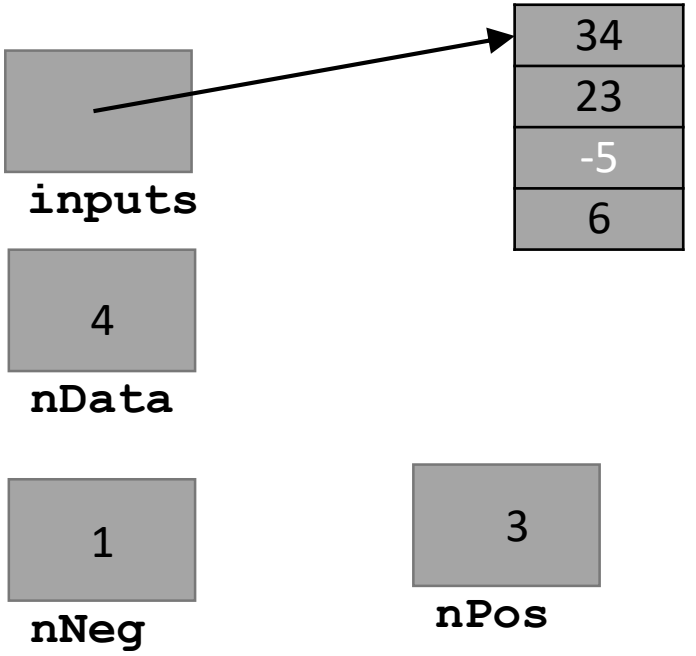


# Remplissage tableaux négatifs et positifs

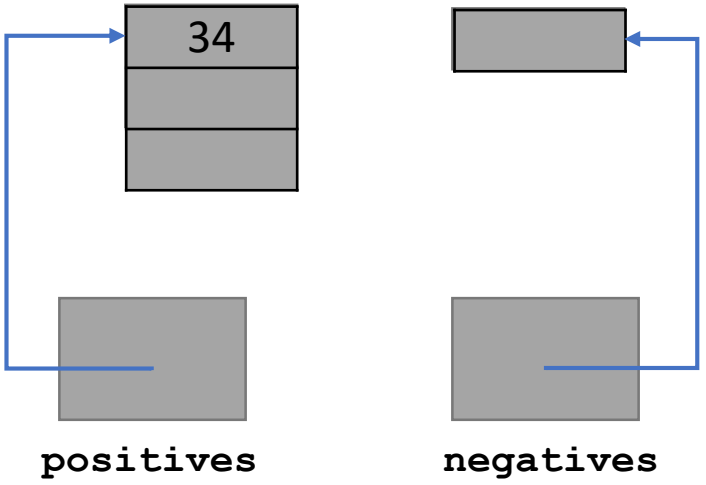
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {
   ...
   // Fill pos and neg arrays
   nPos = nNeg = 0;
   for (int i=0; i<nData; i++) {
      if (inputs[i]>=0) {
         positives[nPos] = inputs[i];
         nPos++;
      } else ...
```

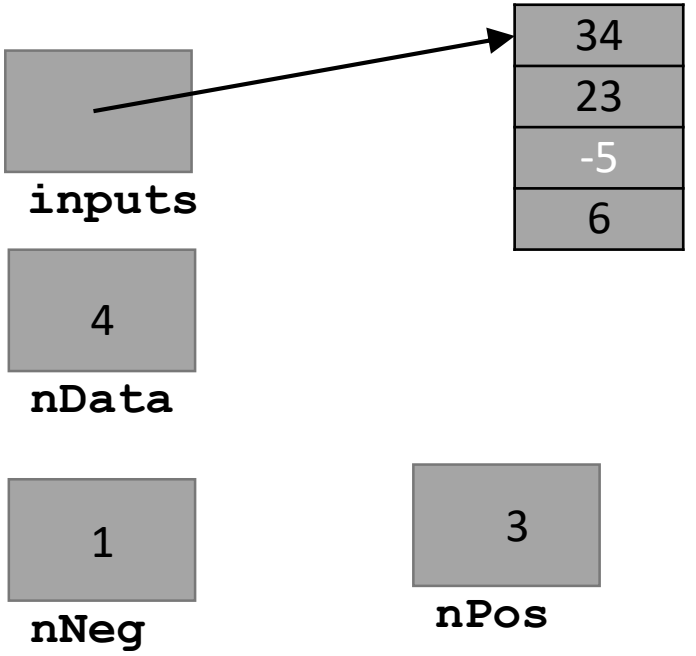


# Remplissage tableaux négatifs et positifs

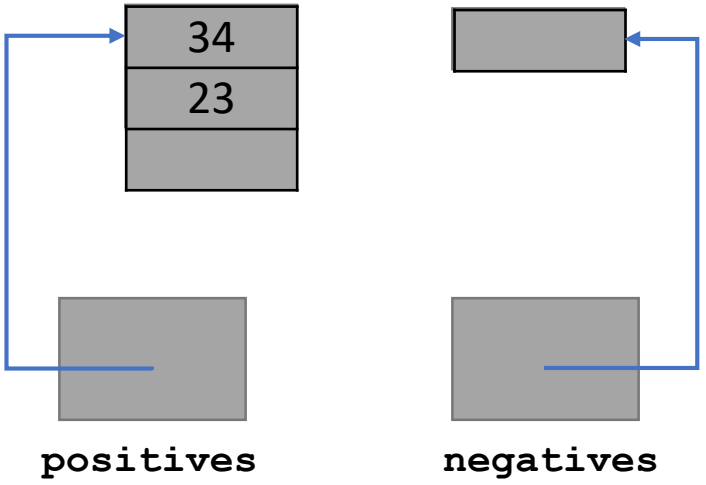
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {
   ...
   // Fill pos and neg arrays
   nPos = nNeg = 0;
   for (int i=0; i<nData; i++) {
      if (inputs[i]>=0) {
         positives[nPos] = inputs[i];
         nPos++;
      } else ...
```

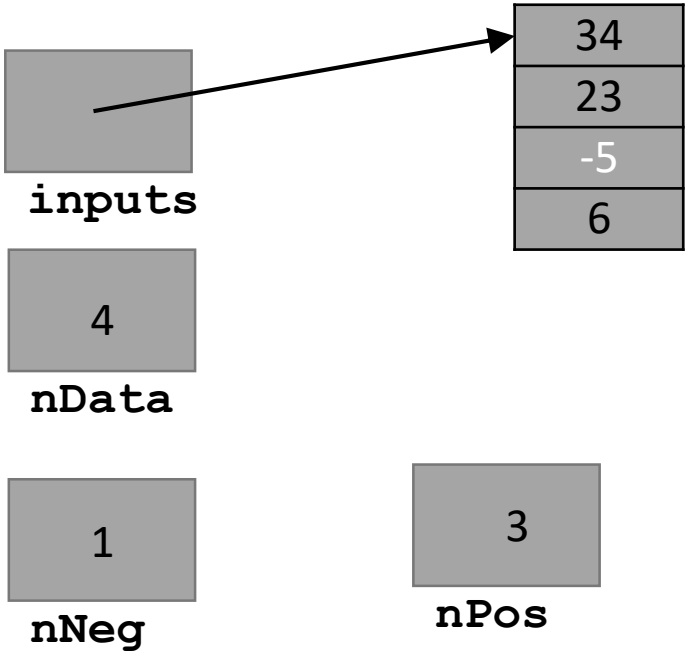


# Remplissage tableaux négatifs et positifs

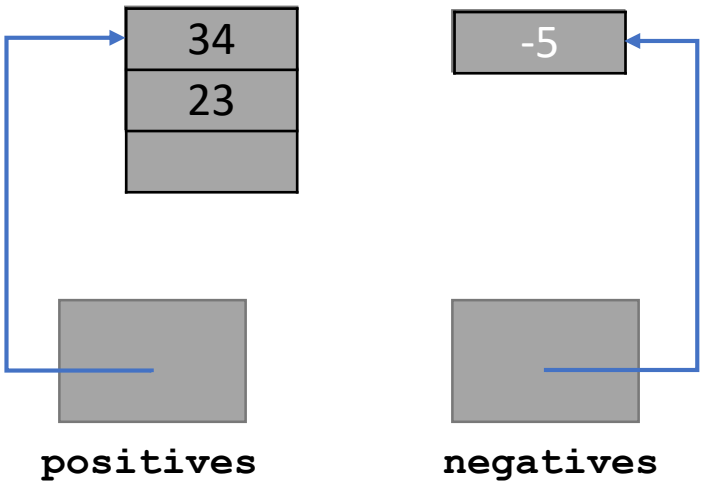
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {
   ...
   // Fill pos and neg arrays
   nPos = nNeg = 0;
   for (int i=0; i<nData; i++) {
      if (inputs[i]>=0) {
         positives[nPos] = inputs[i];
         nPos++;
      } else ...
```

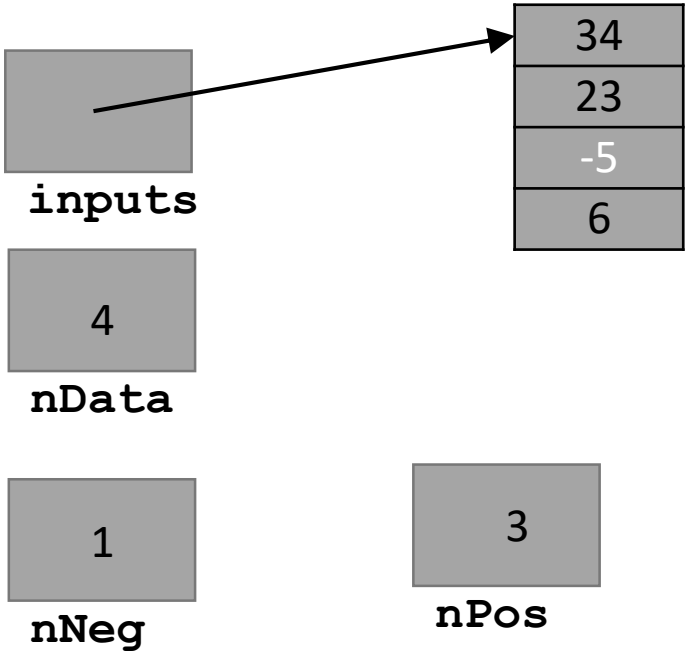


# Remplissage tableaux négatifs et positifs

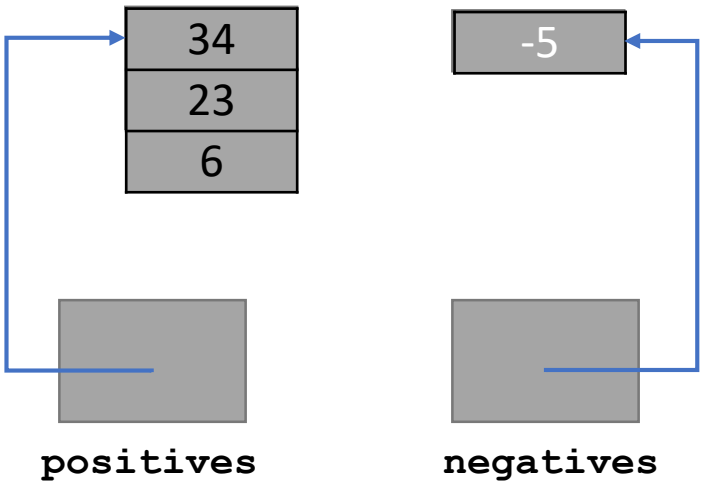
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {
   ...
   // Fill pos and neg arrays
   nPos = nNeg = 0;
   for (int i=0; i<nData; i++) {
      if (inputs[i]>=0) {
         positives[nPos] = inputs[i];
         nPos++;
      } else ...
```

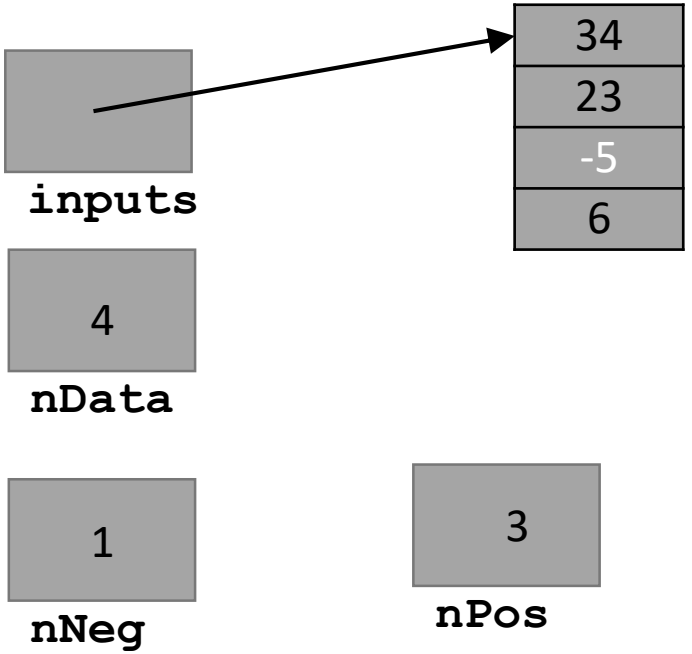


# Remplissage tableaux négatifs et positifs

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

while (nData>0) {
   ...
   // Fill pos and neg arrays
   nPos = nNeg = 0;
   for (int i=0; i<nData; i++) {
      if (inputs[i]>=0) {
         positives[nPos] = inputs[i];
         nPos++;
      } else ...
```



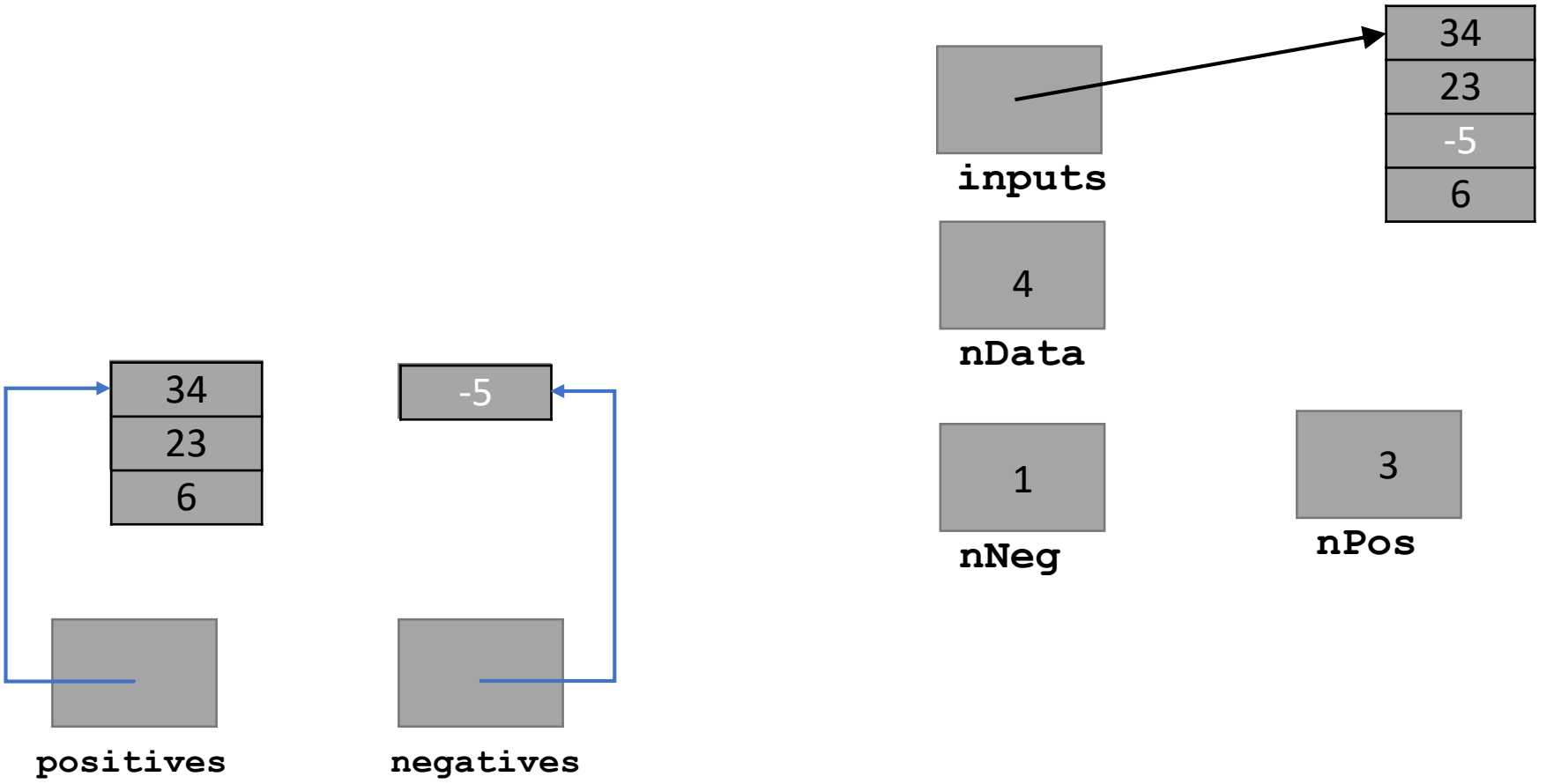



# Phase 3

# Affichage résultats

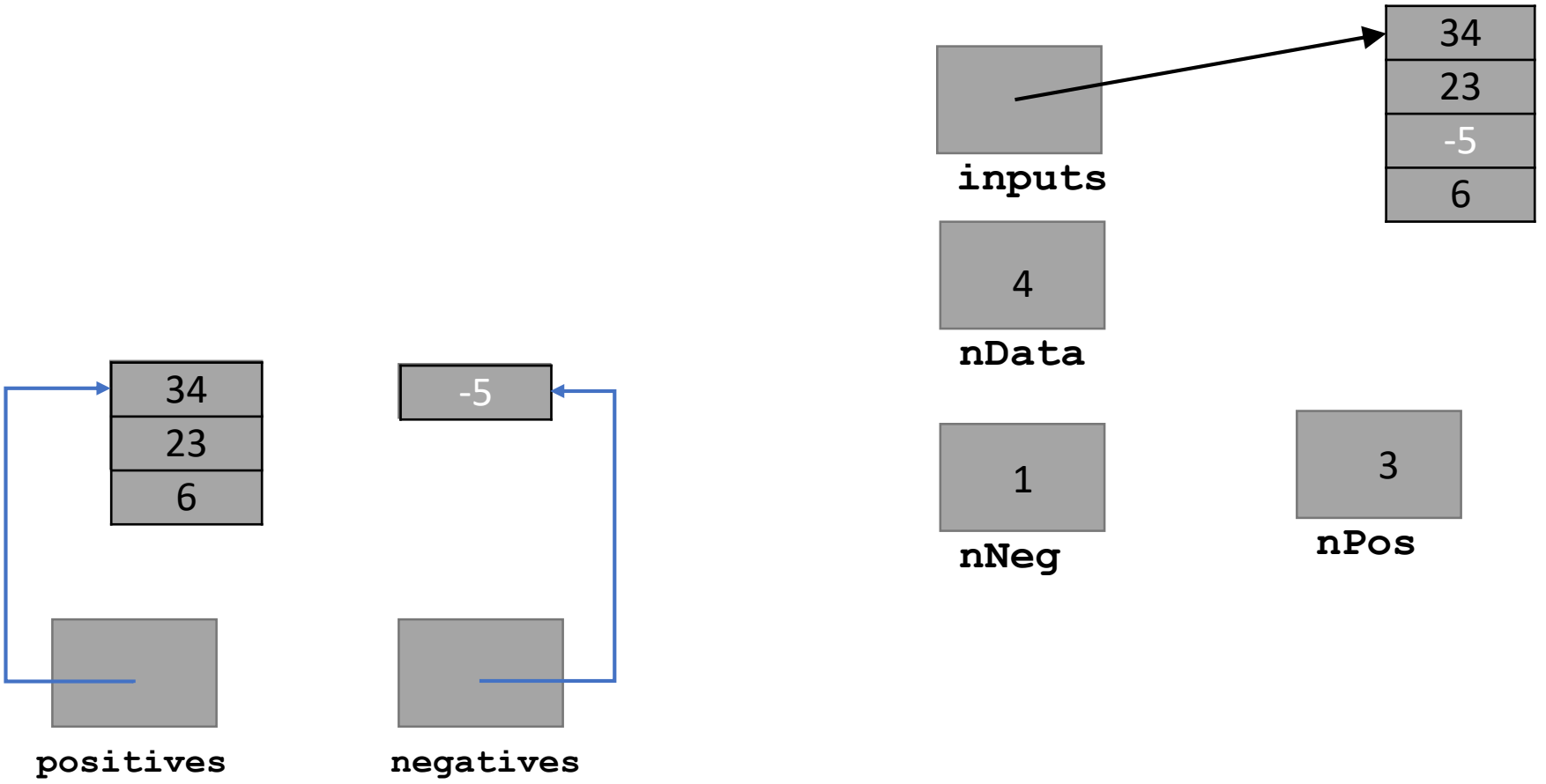
# Affichage résultats

```
for (int i=0; i<nPos; i++)
        printf("%d ", positives[i]);
```



# Affichage résultats

```
for (int i=0; i<nNeg; i++)
        printf("%d ", negatives[i]);
```



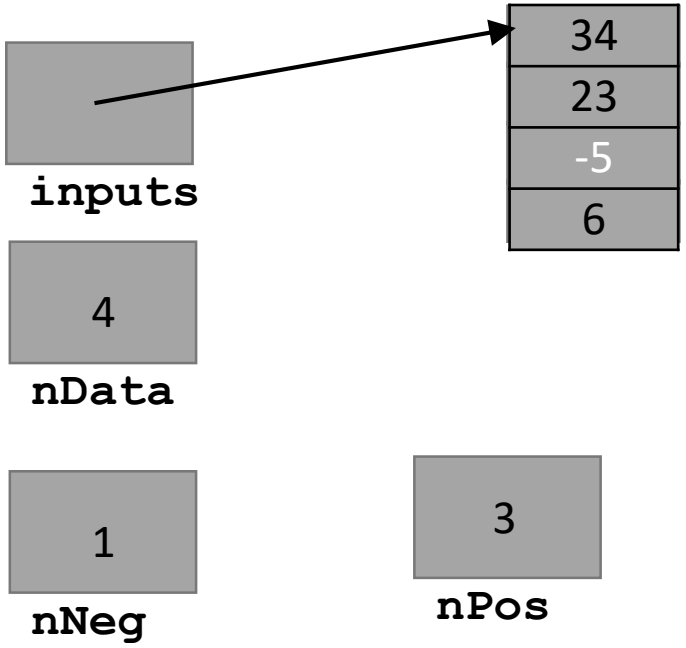
# Phase 4

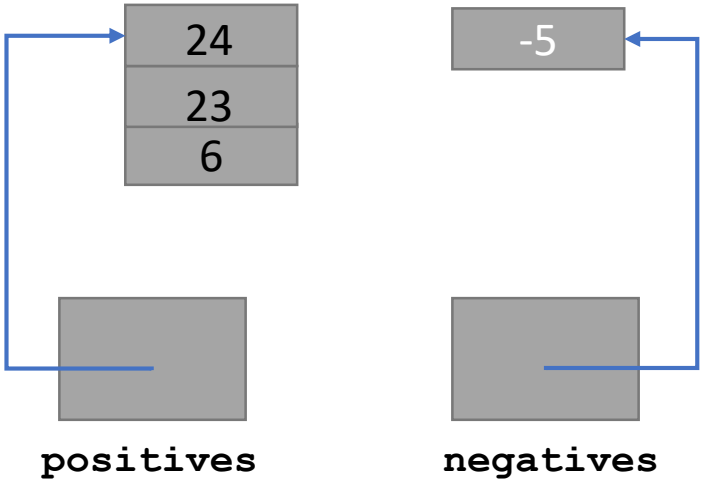
# Libérations tableaux

# 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);
      }
      ...
```

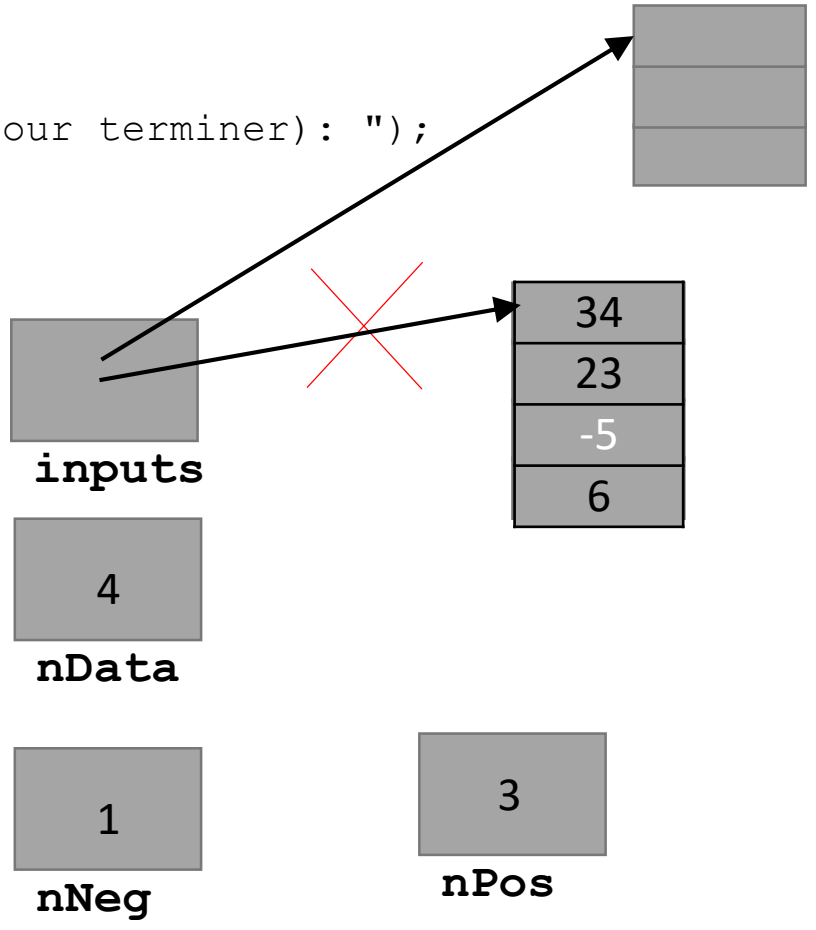
Boucle englobante :
nouveau tableau à chaque itération

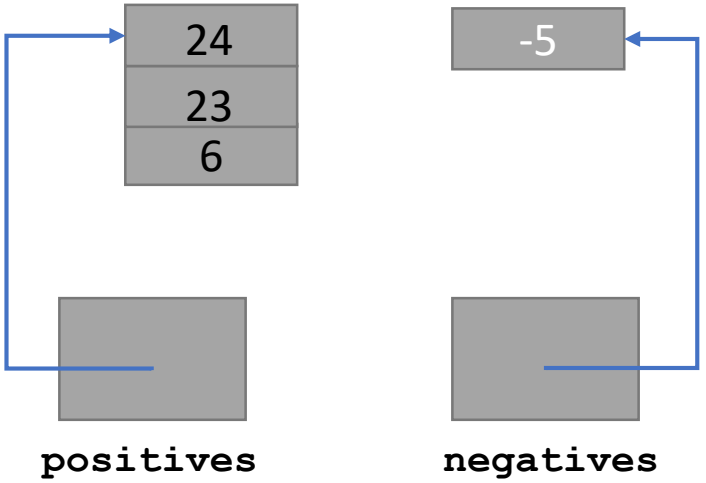


# Libérations tableaux

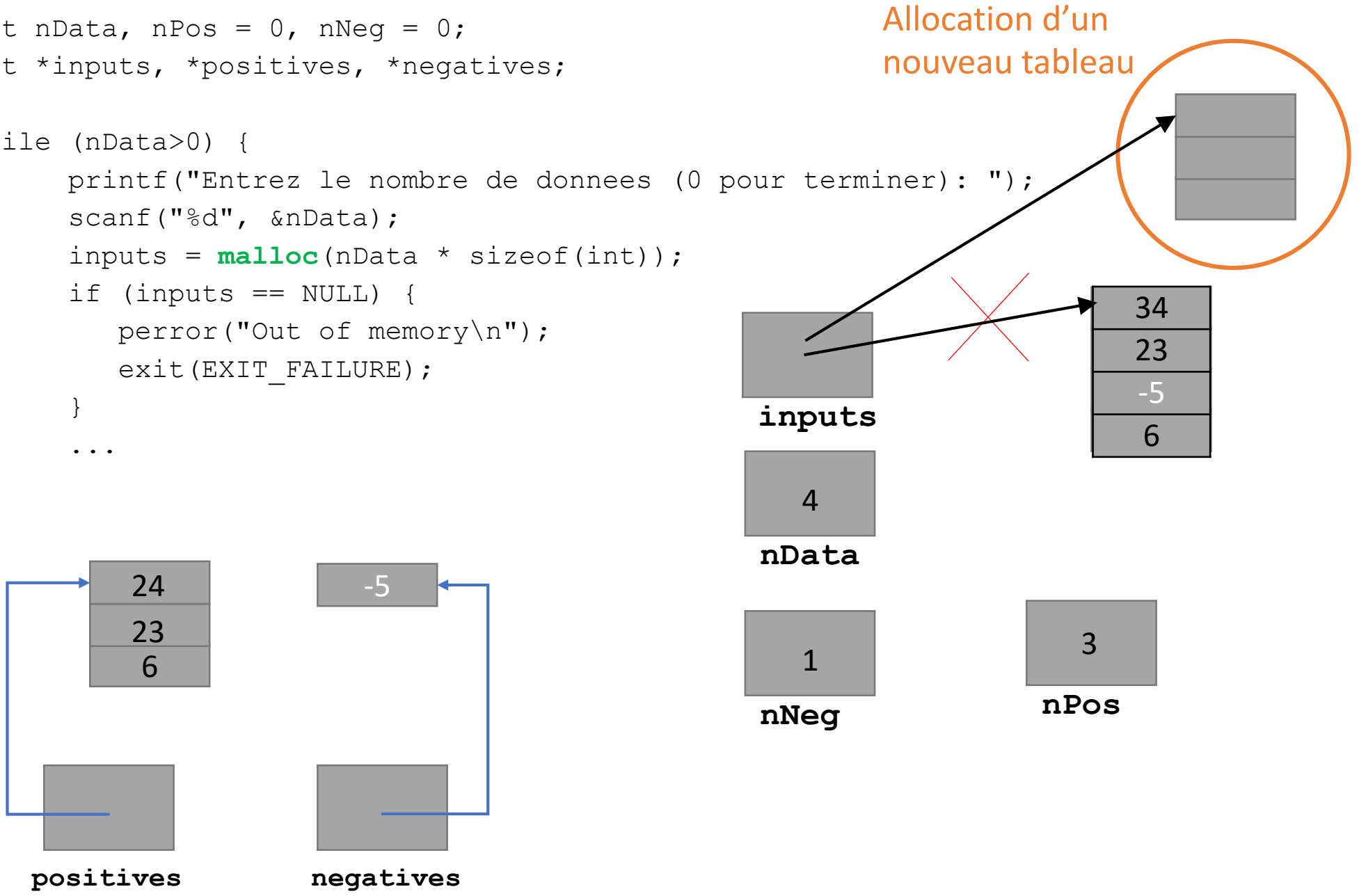
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

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);
      }
      ...
```



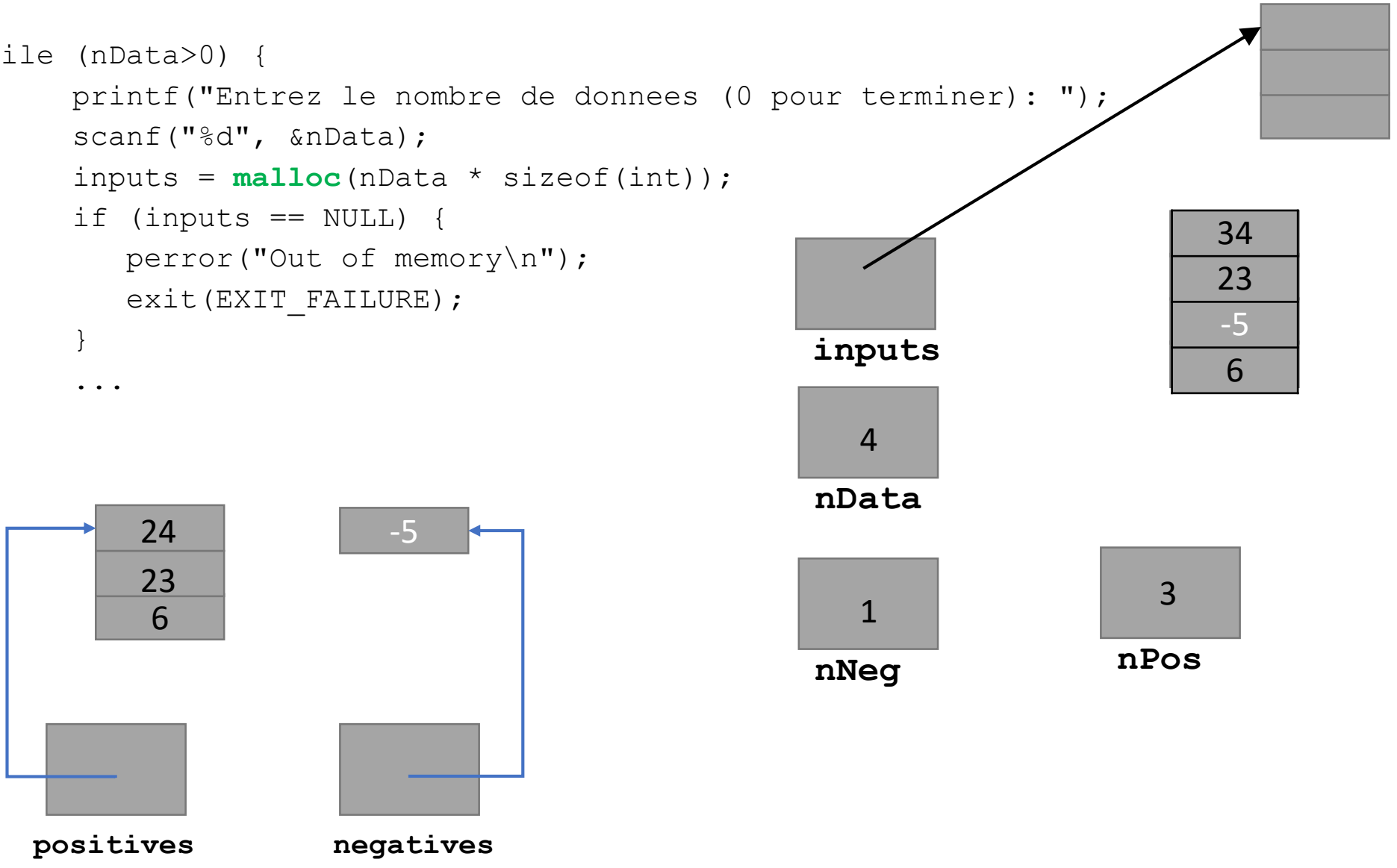
# Libérations tableaux

```
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);
      }
      ...
```



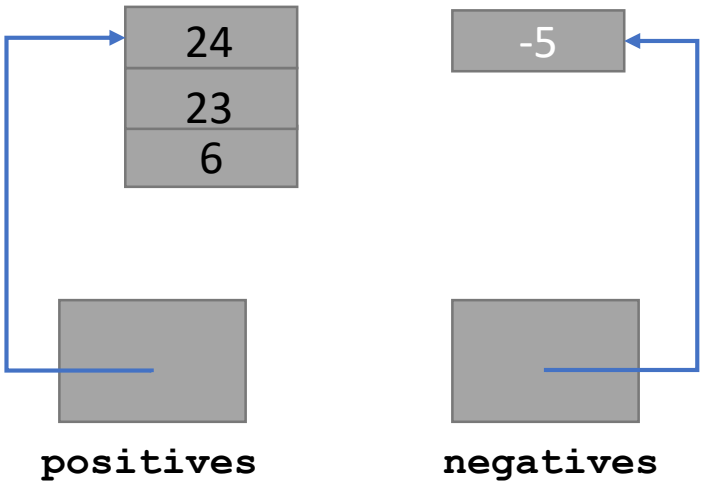
# Libérations tableaux

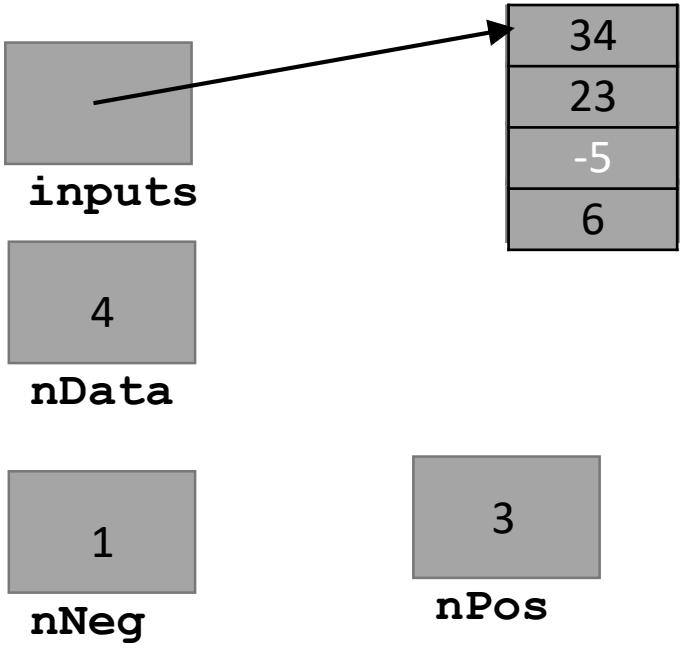
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

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);
}
```

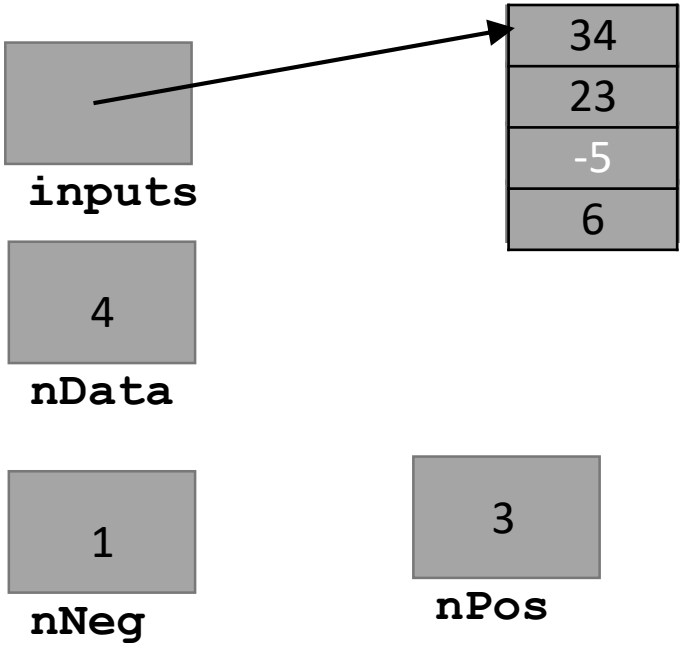


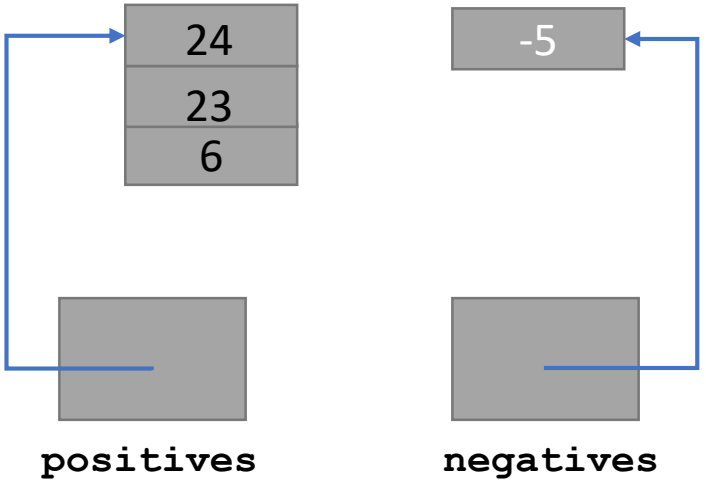



# Libérations tableaux

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

while (nData>0) {

      ...

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

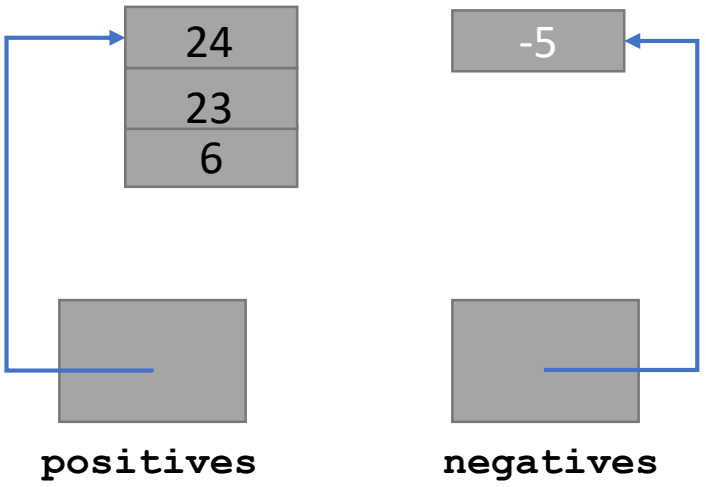
Libération des zones de mémoire dynamique

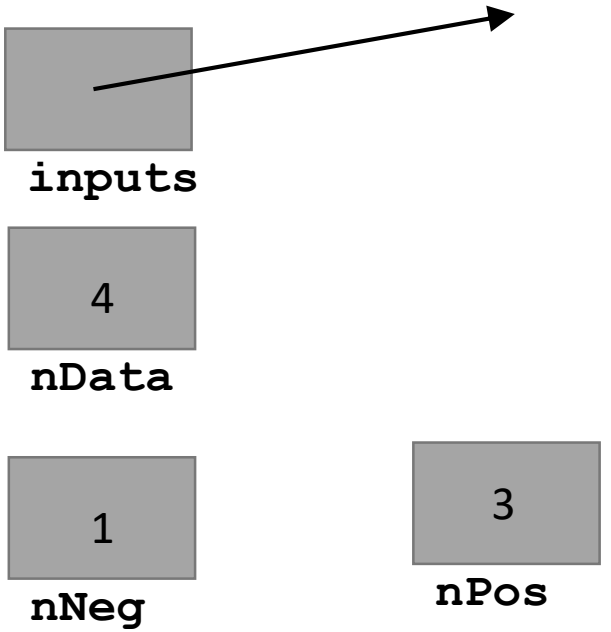


# Libérations tableaux

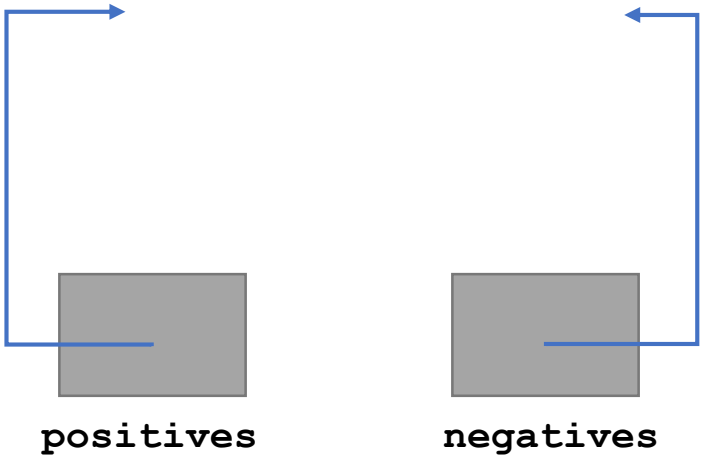
```
int nData, nPos = 0, nNeg = 0;
int *inputs, *positives, *negatives;

while (nData>0) {

      ...

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

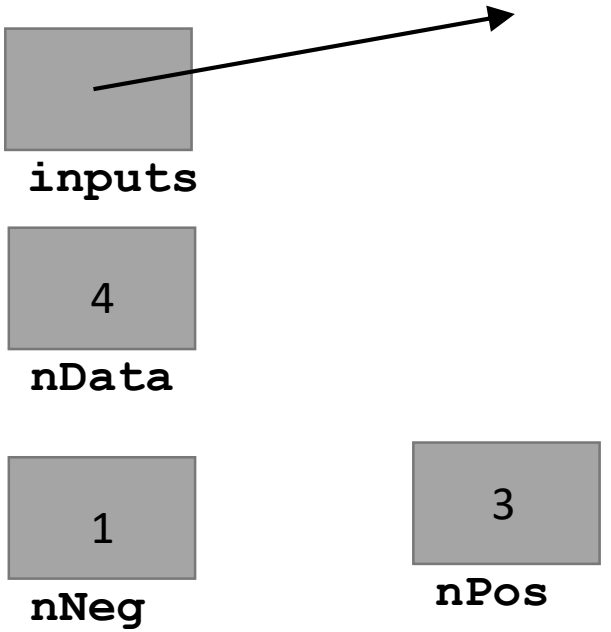


# Libérations tableaux

```
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);
}
```