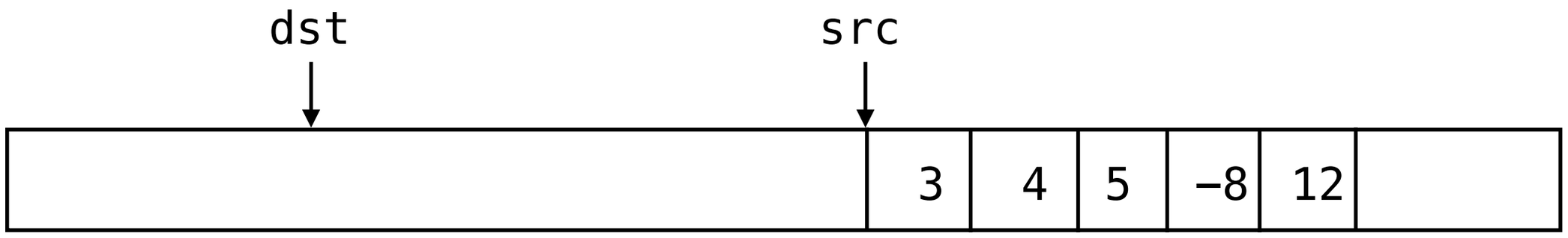
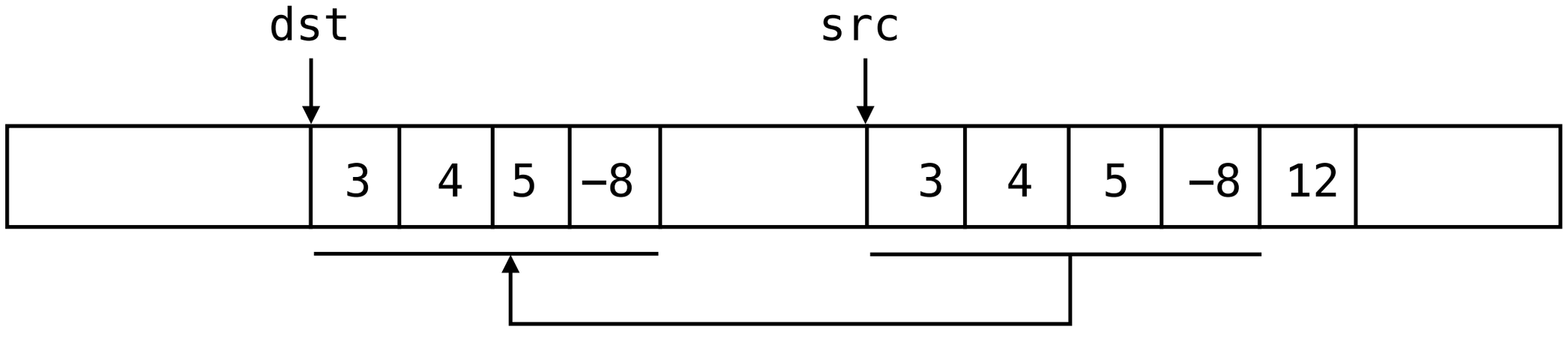
Langage C
exemple : fusion de deux tableaux triés

memmove

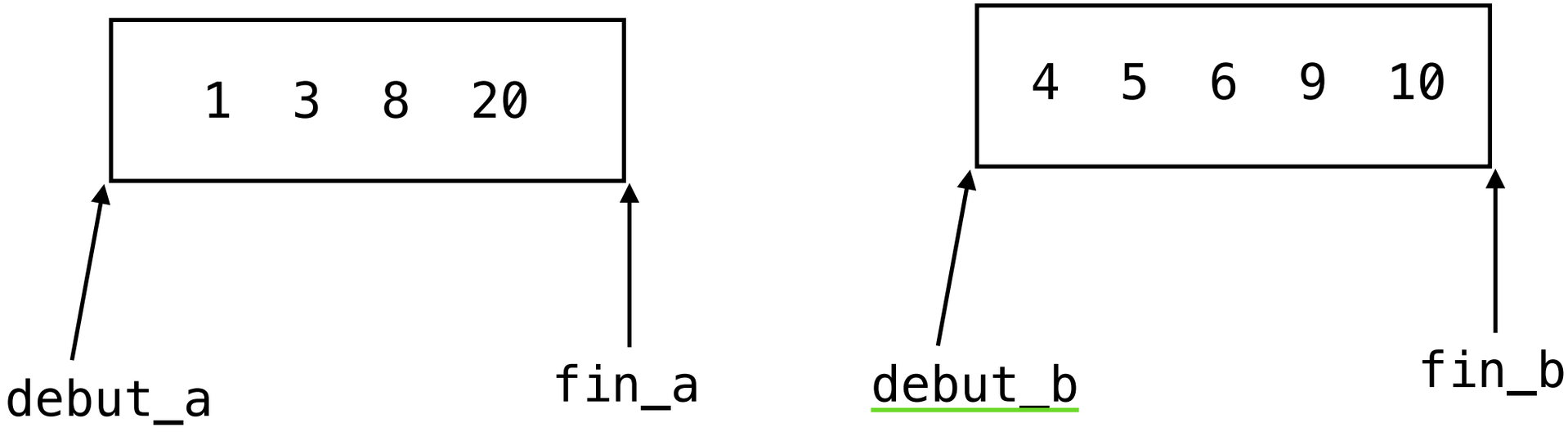
```
#include <string.h>

void *memmove(void *dst, const void *src, size_t n)
```



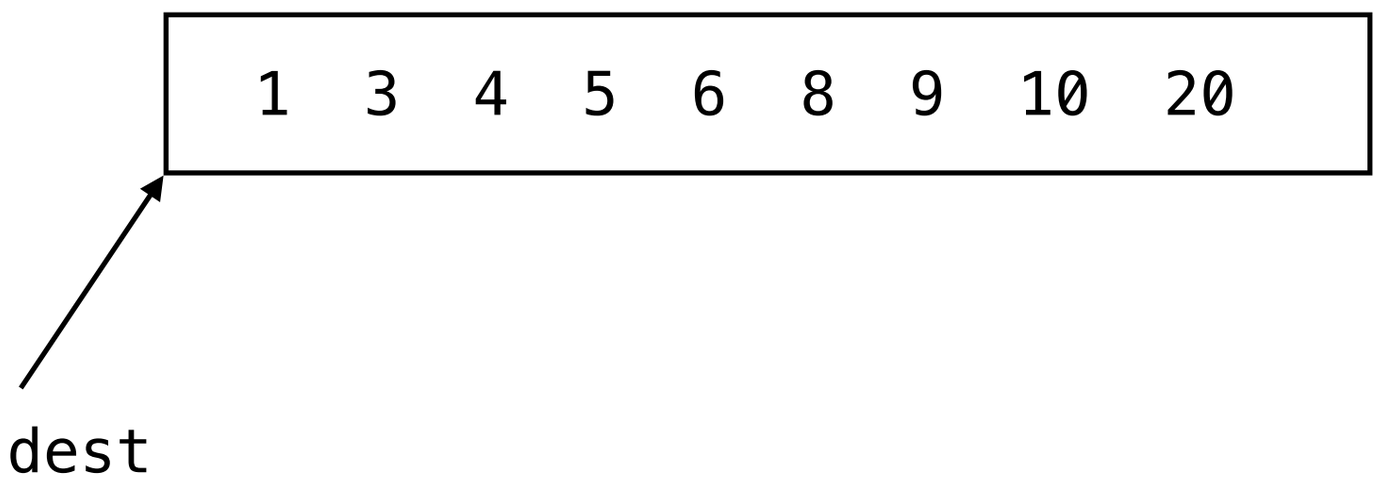

```
memmove( dst, src, 4*sizeof(int) )
```



exemple : fusion de deux tableaux triés de int




```
fusion(int *debut_a, int *fin_a, int *debut_b, int *fin_b, int *dest)
```



exemple : fusion de deux tableaux triés

```
void fusion(int *debut_a, int *fin_a,
            int *debut_b, int *fin_b, int *dest){

  while(debut_a < fin_a && debut_b < fin_b ){
    if( *debut_a <= *debut_b){
       *dest = *debut_a ;
       debut_a++ ;
    }
    else{
       *dest = *debut_b ;
       debut_b++ ;
    }
    dest++;
  }

  if( debut_a < fin_a)
     memmove(dest, debut_a, sizeof(int) * (fin_a - debut_a));
  else
     memmove(dest, debut_b, sizeof(int) * (fin_b - debut_b));
}


int main(){
    int ta[]={-3, -8, 99, 120, 500};
    int tb[]={-100, -2, 40, 155};
    int *tab = malloc( sizeof(ta) + sizeof(tb) );
    assert( tab != NULL );


    fusion(ta, ta+5, tb, tb+4, tab);
    ...........
}
```

exemple : fusion de deux tableaux triés

```
void fusion(int *debut_a, int *fin_a,
            int *debut_b, int *fin_b, int *dest){


  while(debut_a < fin_a && debut_b < fin_b ){
    if( *debut_a <= *debut_b)
       *dest++ = *debut_a++;
    else
       *dest++ = *debut_b++;
  }


  if( debut_a < fin_a)
     memmove(dest, debut_a,  sizeof(int) * (fin_a - debut_a));
  else
     memmove(dest, debut_b,  sizeof(int) * (fin_b - debut_b));
}
```

si a et b pointeurs alors

`*a++ = *b++ ;`

est équivalent à

`*a = *b ;  a++;  b++;`

parce que `*a++ = *b++;`

c'est

`*(a++) = *(b++) ;`

assert(condition)

```
#include <assert.h>

assert(condition);
```

Si la condition est FALSE (s'évalue à 0) alors

- `assert()` envoie sur la sortie standard un message :

assertion failed, nom de la fonction, le nom de fichier source, le numéro de la ligne dans le fichier source et

- et ensuite `assert()` exécute `abort()` ce qui termine le programme.

En définissant la constante NDEBUG avant #include <assert.h> on désactive les assertions:

```
#define NDEBUG

#include <assert.h>
```

Une autre possibilité pour désactiver les assertions ajouter l'option `-DNDEBUG` à la compilation.

```
int i;

.......

assert( i < val  && i>= 0);
```