|  |  |  |
| --- | --- | --- |
| void insertion\_sort(int \*a, int n)  { int i, j, temp;  for (i = 1; i < n; i++)  { j = i;  while (j > 0 && a[j] < a[j - 1])  { swap ( a[j], a[j - 1]);  j--; } } } вставка  void shell\_sort(int \*a, int n)  { double step = n – 1;  while (step >= 1) {  for (int i = 0; i + step < n; ++i) {  if (a[i] > a[i + step]) {  swap(a[i], a[i + step]); } }  step /= 1.247; | void merge\_sort(int \*a, int l, int r)  { int m = (l + r) / 2;  if (l == r) return;  merge\_sort(a, l, m);  merge\_sort(a, m + 1, r);  int i = l, j = m + 1, k = 0;  int \*b;  b = malloc((r - l + 1) \* sizeof(int));  while (i <= m && j <= r)  { if (a[j] < a[i])  { b[k] = a[j];  j++; k++; }  else  { b[k] = a[i];  i++; k++; } }  for (; i <= m; i++, k++) b[k] = a[i];  for (; j <= r; j++, k++) b[k] = a[j];  for (i = l, k = 0; i <= r; i++, k++)  a[i] = b[k];} | Void quick\_sort(int \*a, int l, int r)  { int i = l, j = r, temp, sr = a[(l +r)/2];  do  { while (a[i] < sr) i++;  while (a[j] > sr) j--;  if (i <= j)  { temp = a[j];  a[j] = a[i];  a[i] = temp;  i++;  j--; } } while (i <= j);  if (i < r)  quick\_sort(a, i, r);  if (j > l)  quick\_sort(a, l, j); |
| void mergeSorting(int \*array, int left, int mid, int right)  { int step\_left = 0, step\_right = 0;  int \*result = (int \*)malloc((right - left + 1) \* sizeof(int));  while (left + step\_left < mid && mid + step\_right < right)  { if (array[left + step\_left] < array[mid + step\_right])  { result[step\_left + step\_right] = array[left + step\_left];  step\_left++; }  else { result[step\_left + step\_right] = array[mid + step\_right];  step\_right++; } }  while (left + step\_left < mid)  { result[step\_left + step\_right] = array[left + step\_left];  step\_left++; } | while (mid + step\_right < right)  { result[step\_left + step\_right] = array[mid + step\_right];  step\_right++; }  for (int i = 0; i < step\_left + step\_right; i++)  array[left + i] = result[i];  free(result);}  void mergeSortingRecursive(int \*array, int left, int right)  { if (left + 1 >= right) return;  int mid = (left + right) / 2;  mergeSortingRecursive(array, left, mid);  mergeSortingRecursive(array, mid, right);  mergeSorting(array, left, mid, right);} |  |
|  |  |  |