Formula

precondition	statement	postcondition
{data = \old(data)}	statement	{containsOldElements(data, \old(data)) & sorted(data)}

Variables
LOCAL int tmp
LOCAL int j
LOCAL int i

Global Conditions
data != null
i >= 0 & i <= data.length
j <= data.length-2 & j >=i-1

Statement1

precondi...	statement	postcond...
{data = \old(data)}	i = 0;	{i = 0}

Composition

precondition	postcondition
{data = \old(data)}	{containsOldElements(data, \old(data)) & sorted(data)}

statement 1	intermediate condition	statement 2
statement1	{i = 0}	statement2

RepetitionStatement1

RepetitionStatement DO...OD

invariant	guard	variant
(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m]))))	i < data.length	data.length-i

precondition	loop statement	postcondition
{((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (i < data.length)}	loop	{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (i = data.length)}

Composition

precondition	postcondition
{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & j = data.length-2}	{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m]))))}

statement 1	intermediate condition	statement 2
statement1	{(\forall int k; (0 <= k & k < i+1 -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m]))))}	statement2

Composition

precondition	postcondition
{((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))) & (j>=i)}	{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))}

statement 1	intermediate condition	statement 2
statement1	{((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))) & (j>=i)}	statement2

Statement2

precondition	statement	postcondition
{((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (i < data.length)}	j = data.length-2;	{((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & j = data.length-2}

Statement5

precondition	statement	postcondition
{(\forall int k; (0 <= k & k < i+1 -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m]))))}	i++;	{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m]))))}

RepetitionStatement2

RepetitionStatement DO...OD

invariant	guard	variant
(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))	j>=i	j+1

precondition	loop statement	postcondition
{((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))) & (j>=i)}	loop	{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))}

Composition

precondition	postcondition
{((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))) & (j>=i)}	{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))}

statement 1	intermediate condition	statement 2
statement1	{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j] <= data[h]))}	statement2

SelectionStatement1

SelectionStatement IF..FI

guards	
data[j] > data[j+1]	data[j] <= data[j+1]

precondition	
{modifiable(\nothing);(((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))) & (j>=i)) & (data[j] > data[j+1])}	{modifiable(\nothing);(((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))) & (j>=i)) & (data[j] <= data[j+1])}

statements	
statement	statement

postcondition
{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j] <= data[h]))}

Statement4

precondition	statement	postcondition
{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j] <= data[h]))}	j--;	{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))}

Statement3

precondition	statement	postcondition
{modifiable(\nothing);(((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))) & (j>=i)) & (data[j] > data[j+1])}	tmp = data[j]; data[j] = data[j+1]; data[j+1] = tmp;	{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j] <= data[h]))}

SkipStatement1

Skip

precondition	postcondition
{modifiable(\nothing);(((\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j+1] <= data[h]))) & (j>=i)) & (data[j] <= data[j+1])}	{(\forall int k; (0 <= k & k < i -> (\forall int m; (k < m & m < data.length -> data[k] <= data[m])))) & (\forall int h; (j < h & h < data.length -> data[j] <= data[h]))}