

```

1  int pivot(int A[], int i, int j)
2  {
3      int k = i+1;
4
5      while(k <= j && A[i] == A[k]) k++;
6      if(k > j) return -1;      /* A[i]~A[j] がすべて同じ値の場合 */
7      else if(A[i] >= A[k]) return i; /* 軸となる要素の添字を返す */
8      else return k;
9  }

```

図 7: 軸要素選択のプログラム (プリントの図 4.E を少し修正してあるが, アルゴリズムとしては図 4.E と同じ)

```

1  int partition(int A[], int i, int j, int a)
2  /* a を基準として A[i]~A[j] を 2 つのグループ (a 未満 A[i]~A[k-1] と */
3  /* a 以上 A[k]~A[j] のグループ) に分割して, k を返す */
4  {
5      int L = i, R = j;
6
7      while(A[L] < a) L++;
8      while(A[R] >= a) R--;
9      while(L <= R){
10         swap(A,L,R);
11         L++, R--;
12         while(A[L] < a) L++;
13         while(A[R] >= a) R--;
14     }
15     return L;
16 }

```

図 8: 分割のプログラム (プリントの図 4.F と同じ)

```

1  void quicksort(int A[], int i, int j)
2  {
3      int a, k, p;
4
5      p = pivot(A,i,j); /* 軸要素 A[p] を選択 */
6      if(p != -1){ /* A[i]~A[j] がすべて等しい (または i = j) ときには何もしない */
7          a = A[p];
8          k = partition(A,i,j,a); /* 軸要素 a を用いて, 配列 A を分割 */
9          quicksort(A,i,k-1); /* A[i] ~ A[k-1] に対してクイックソートを実行 */
10         quicksort(A,k,j); /* A[k] ~ A[j] に対してクイックソートを実行 */
11     }
12 }

```

図 9: クイックソートのプログラム (プリントの図 4.G と同じ)