Script started on Wed Nov 20 10:35:36 2019

[?1034hbash-3.2$ cat radix.cpp

#include <iostream>

#include <cstdlib>

#include <queue>

#include <vector>

#include <string>

#include <math.h>

using namespace std;

int numDigits(int k);

int maxElemLength(const vector<int> &v); //Return # of digits of largest int in v

int GetDigit(int number, int k); // Return the kth digit of number

vector< queue<int> > ItemsToQueues(const vector<int> &L, int k); // create and return intermediate array of ten queues

vector<int> QueuesToArray(vector< queue<int> > &QA); // create and return new list from values of intermediate array of queues

void RadixSort(vector<int> &L, int numDigits); // calls both functions above to sort the list

void PrintVector(const vector<int> &L); // prints contents of vector with no modification

int main()

{

vector<int> L;

L.push\_back(380);

L.push\_back(95);

L.push\_back(345);

L.push\_back(382);

L.push\_back(260);

L.push\_back(100);

cout << "\nUnsorted Array: ";

PrintVector(L);

RadixSort(L, maxElemLength(L));

cout << "\n\nSorted Array: ";

PrintVector(L);

cout << "\n" << endl;

return 0;

}

int numDigits(int k)

{

int power = 0;

while((k - pow(10, power)) > 0)

power++;

return power;

}

int maxElemLength(const vector<int> &v)

{

int biggest = 0;

for (int i=0; i < v.size(); i++)

{

if (v.at(i) >= biggest)

biggest = v.at(i);

}

return numDigits(biggest);

}

int GetDigit(int number, int k)

{

//int numDigit = numDigits(number);

int x = number / (pow(10, k));

int y = (pow(10, k)) \* x;

int z = number - y;

return z / (pow(10, k-1));

}

vector< queue<int> > ItemsToQueues(const vector<int> &L, int k)

{

vector< queue<int> > step1;

for (int i = 0; i < 10; i++)

{

queue<int> QA;

for (int j=0; j < L.size(); j++)

{

if(GetDigit(L.at(j), k) == i)

{

QA.push(L.at(j));

}

}

step1.push\_back(QA);

}

return step1;

}

vector<int> QueuesToArray(vector< queue<int> > &QA)

{

vector<int> step2;

for (int i=0; i < QA.size(); i++)

{

while(!QA.at(i).empty())

{

step2.push\_back(QA.at(i).front());

QA.at(i).pop();

}

}

return step2;

}

void RadixSort(vector<int> &L, int numDigits)

{

for (int i = 1; i <= numDigits; i++)

{

vector< queue<int> > step = ItemsToQueues(L, i);

L = QueuesToArray(step);

}

}

void PrintVector(const vector<int> &L)

{

for (int i=0; i < L.size(); i++)

{

cout << L.at(i) << " ";

}

}

bash-3.2$ c++ radix.cpp

bash-3.2$ ./a.out

Unsorted Array: 380 95 345 382 260 100

Sorted Array: 95 100 260 345 380 382

bash-3.2$ c++[K[Kat radixS.cpp

#include <iostream>

#include <cstdlib>

#include <queue>

#include <vector>

#include <string>

#include <math.h>

using namespace std;

int maxElemLength(const vector<string> &v); //Return # of chars of largest string in v

char GetChar(string entry, int k); // Return the kth char of entry

vector< queue<string> > ItemsToQueues(const vector<string> &L, int k); // create and return intermediate array of ten queues

vector<string> QueuesToArray(vector< queue<string> > &QA); // create and return new list from values of intermediate array of queues

void RadixSort(vector<string> &L, int numDigits); // calls both functions above to sort the list

void PrintVector(const vector<string> &L); // prints contents of vector with no modification

void PadStrings(vector<string> &L, int maxLength);

void UnPadStrings(vector<string> &L);

int main()

{

vector<string> L;

string a[] = {"zebra", "apple", "orange", "can", "candy", "a",

"top", "pumpkin", "today", "parade"};

int n = sizeof(a) / sizeof(a[0]);

for (int i = 0; i < n; i++)

{

L.push\_back(a[i]);

}

cout << "\nUnsorted Array: ";

PrintVector(L);

int big = maxElemLength(L);

PadStrings(L, big);

RadixSort(L, big);

UnPadStrings(L);

cout << "\n\nSorted Array: ";

PrintVector(L);

cout << "\n" << endl;

return 0;

}

int maxElemLength(const vector<string> &v)

{

int biggest = 0;

for (int i=0; i < v.size(); i++)

{

if (v.at(i).size() >= biggest)

biggest = v.at(i).size();

}

return biggest;

}

void PadStrings(vector<string> &L, int maxLength)

{

for (int i = 0; i < L.size(); i++)

{

L.at(i).append(maxLength - L.at(i).length(), ' ');

}

}

void UnPadStrings(vector<string> &L)

{

for (int i = 0; i < L.size(); i++)

{

while(L.at(i).back() == ' ')

{

L.at(i).pop\_back();

}

}

}

char GetChar(string entry, int k)

{

return entry[k];

}

vector< queue<string> > ItemsToQueues(const vector<string> &L, int k)

{

vector< queue<string> > step1;

for (int i = 0; i < 128; i++)

{

queue<string> QA;

for (int j=0; j < L.size(); j++)

{

if(GetChar(L.at(j), k) == i)

{

QA.push(L.at(j));

}

}

step1.push\_back(QA);

}

return step1;

}

vector<string> QueuesToArray(vector< queue<string> > &QA)

{

vector<string> step2;

for (int i=0; i < QA.size(); i++)

{

while(!QA.at(i).empty())

{

step2.push\_back(QA.at(i).front());

QA.at(i).pop();

}

}

return step2;

}

void RadixSort(vector<string> &L, int numDigits)

{

for (int i = numDigits; i >= 0; i--)

{

vector< queue<string> > step = ItemsToQueues(L, i);

L = QueuesToArray(step);

}

}

void PrintVector(const vector<string> &L)

{

for (int i=0; i < L.size(); i++)

{

cout << L.at(i) << " ";

}

}

bash-3.2$ c++ radis[Kc[Kxs[KS.cpp

bash-3.2$ ./a.out

Unsorted Array: zebra apple orange can candy a top pumpkin today parade

Sorted Array: a apple can candy orange parade pumpkin today top zebra

bash-3.2$ exit

exit

Script done on Wed Nov 20 10:36:38 2019