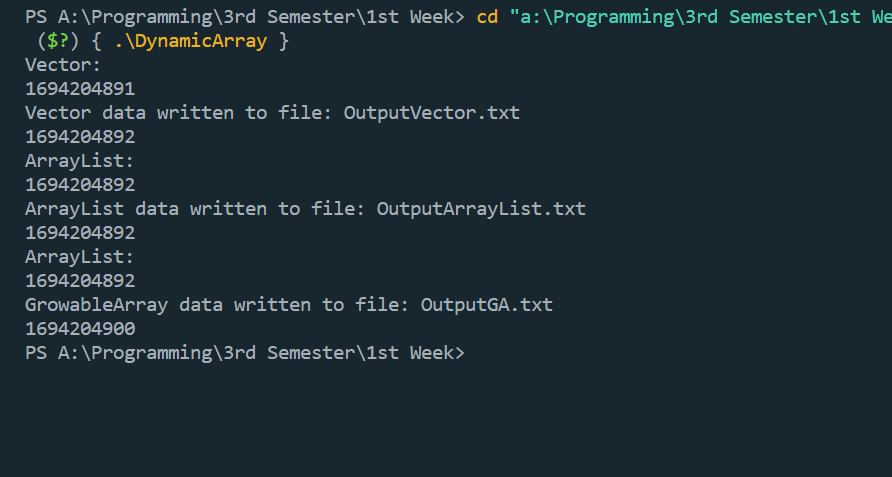
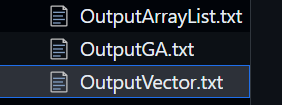
Lists backend implementation





Here is the code:

#include <ostream>

#include "vector.cpp"

#include <fstream>

#include <time.h>

#include <bits/stdc++.h>

#include <string>

using namespace std;

template <typename T>

class DynamicArray

{

public:

    int size, capacity;

    T \*arr;

    DynamicArray()

    {

        size = 0, capacity = 1;

        arr = new T[capacity];

    }

    void append(T value)

    {

        if (capacity == size)

        {

            capacity += 1;

            T \*newArr = new T[capacity];

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

            {

                newArr[i] = arr[i];

            }

            arr = newArr;

        }

        arr[size] = value;

        size++;

    };

    T &operator[](int i)

    {

        return arr[i];

    }

    T pop()

    {

        size--;

        return arr[size];

    };

    int len()

    {

        return size;

    };

};

template <typename T>

ostream &operator<<(ostream &cout, DynamicArray<T> &list)

{

    for (int i = 0; i < list.len(); i++)

    {

        cout << list[i] << endl;

    }

    return cout;

}

template <typename T>

class ArrayList

{

public:

    int size, capacity;

    T \*arr;

    ArrayList()

    {

        size = 0, capacity = 2;

        arr = new T[capacity];

    }

    void append(T value)

    {

        if (capacity == size)

        {

            capacity \*= 1.5;

            T \*newArr = new T[capacity];

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

            {

                newArr[i] = arr[i];

            }

            arr = newArr;

        }

        arr[size] = value;

        size++;

    };

    T &operator[](int i)

    {

        return arr[i];

    }

    T pop()

    {

        size--;

        return arr[size];

    };

    int len()

    {

        return size;

    };

};

template <typename T>

ostream &operator<<(ostream &cout, ArrayList<T> &list)

{

    for (int i = 0; i < list.len(); i++)

    {

        cout << list[i] << endl;

    }

    return cout;

}

// creating an output file stream

void CreateFile(string path)

{

    srand(time(0));

    ofstream writer(path);

    for (int i = 0; i < 1024 \* 1024; i++)

    {

        writer << rand() % 10 << " ";

    }

}

// reading from file for Vector

template <typename T>

void readFile(string path, Vector<T> &list)

{

    ifstream read(path);

    string line;

    getline(read, line);

    for (int i = 0; i < line.length(); i += 2)

    {

        list.append(line[i]);

    }

}

// writing to file for Vector

template <typename T>

void writeFile(string path, Vector<T> &list)

{

    ofstream writer(path);

    for (int i = 0; i < list.len(); i++)

    {

        writer << list[i] << " ";

    }

    cout << "Vector data written to file: " << path << endl;

}

// reading from file for ArrayList

template <typename T>

void readFile(string path, ArrayList<T> &list)

{

    ifstream read(path);

    string line;

    getline(read, line);

    for (int i = 0; i < line.length(); i += 2)

    {

        list.append(line[i]);

    }

}

// writing to file for ArrayList

template <typename T>

void writeFile(string path, ArrayList<T> &list)

{

    ofstream writer(path);

    for (int i = 0; i < list.len(); i++)

    {

        writer << list[i] << " ";

    }

    cout << "ArrayList data written to file: " << path << endl;

}

// reading from file for DynamicArray

template <typename T>

void readFile(string path, DynamicArray<T> &list)

{

    ifstream read(path);

    string line;

    getline(read, line);

    for (int i = 0; i < line.length(); i += 34)

    {

        list.append(line[i]);

    }

}

// writing to file for DynamicArray

template <typename T>

void writeFile(string path, DynamicArray<T> &list)

{

    ofstream writer(path);

    for (int i = 0; i < list.len(); i++)

    {

        writer << list[i] << " ";

    }

    cout << "GrowableArray data written to file: " << path << endl;

}

main()

{

    CreateFile("data.txt");

    // vector

    time\_t start, end;

    cout << "Vector: " << endl;

    cout << time(&start) << endl;

    Vector<char> str;

    readFile("data.txt", str);

    writeFile("OutputVector.txt", str);

    cout << time(&end) << endl;

    // ArrayList

    cout << "ArrayList: " << endl;

    cout << time(&start) << endl;

    ArrayList<char> str2;

    readFile("data.txt", str2);

    writeFile("OutputArrayList.txt", str2);

    cout << time(&end) << endl;

    // DynamicArray

    cout << "ArrayList: " << endl;

    cout << time(&start) << endl;

    DynamicArray<char> str3;

    readFile("data.txt", str3);

    writeFile("OutputGA.txt", str3);

    cout << time(&end) << endl;

}

#include <iostream>

using namespace std;

template<typename T>

class Vector{

    public:

        int size, capacity;

        T\* vector;

        Vector(){

            size = 0, capacity = 1;

            vector = new T[capacity];

        };

        void setCapacity(){

            capacity \*= 2;

        }

        void append(T value){

            if (capacity == size){

                setCapacity();

                T\* newVector = new T[capacity];

                for (int i = 0; i < size; i++){

                    newVector[i] = vector[i];

                }

                vector = newVector;

            }

            vector[size] = value;

            size++;

        };

        T operator[](int i){

            return vector[i];

        }

        T pop(){

            size--;

            return vector[size];

        };

        int len(){

            return size;

        };

};

template <typename T>

ostream& operator<<(ostream &cout, Vector<T> &list)

{

    for (int i = 0; i < list.len(); i++)

    {

        cout << list[i] << endl;

    }

    return cout;

}