**Expandable Array – C++**

#include <iostream>

using namespace *std*;

template <class T>

class EArray

{

private:

T\* array;

int size;

public:

EArray(int l);

~EArray();

void set\_element(int i, const T & newval);

void erase\_element(int i);

void add\_element(int i, const T & newval);

void push(const T & newval);

void output();

};

template <class T>

EArray<T>::EArray(int l)

{

size = l;

array = new T[size];

}

template <class T>

EArray<T>::~EArray()

{

delete[] array;

array = *NULL*;

}

template <class T>

void EArray<T>::set\_element(int i, const T & newval)

{

if (i >= 0 && i < size)

{

array[i] = newval;

}

}

template <class T>

void EArray<T>::erase\_element(int index)

{

size -= 1;

T\* newArray = new T[size];

for (int i = 0; i < size+1; i++)

{

if (i < index)

{

newArray[i] = array[i];

}

else if (i > index)

{

newArray[i-1] = array[i];

}

}

delete[] array;

array = newArray;

};

template<class T>

void EArray<T>::add\_element(int index, const T & newval)

{

if (index < size + 1 && index > 0)

{

size += 1;

T\* newArray = new T[size];

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

{

if (i < index)

{

newArray[i] = array[i];

}

else if (i == index)

{

newArray[i] = newval;

}

else

{

newArray[i] = array[i - 1];

}

}

delete[] array;

array = newArray;

}

}

template <class T>

void EArray<T>::push(const T & newval)

{

size += 1;

T\* newArray = new T[size];

for (int i = 0; i < size - 1; i++)

{

newArray[i] = array[i];

}

newArray[size - 1] = newval;

delete[] array;

array = newArray;

}

template <class T>

void EArray<T>::output()

{

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

{

*std*::*cout* << array[i] << *std*::*endl*;

}

}

//// \*\*\*\*\*\*\*\*\*\* MAIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()

{

EArray <int> myArray(3);

myArray.set\_element(0, 3);

myArray.set\_element(1, 5);

myArray.set\_element(2, 7);

myArray.output();

*std*::*cout* << "---" << *std*::*endl*;

myArray.erase\_element(1);

myArray.output();

*std*::*cout* << "---" << *std*::*endl*;

myArray.push(9);

myArray.output();

*std*::*cout* << "---" << *std*::*endl*;

myArray.add\_element(1, 100);

myArray.output();

*std*::*cout* << "---" << *std*::*endl*;

return 0;

}