1. **Task**

Static unlinked linear list to design on based array. The project must fulfill the following functions- insert of element on the list;

delete of element from the list;

search of element in a static list;

search of position of element in a static list;

print list.

**2. Requirements:**

Type of Input must be an integer.

Limit the size of an array of 100 elements.

1. **Design of diagrams.**

**Design** - The structural method – decomposition - is applied to the solution of the problem. To decompose the task at the top level (modules) and the lowest level (function). Process data for exception cases.

**First level design by top-down**



**Diagrams of the lower-level design**



1. **Testing task**

**Assembling task from the bottom up**

Blocks Input \ Output testing on correct input /output data



**We use stub of unused functions for the correct software code debugging and testing.**

Block (Deleete) are testing for correctness remove an item from list - need to remove an item that is located after the specified index



**Testing all nask**



**Conclution**

While creating program, we learned:

-how to create list in C++ language;

-how to work with list;

-how to work with usual list functions;

-how to create and use switch function.

#include <iostream>

#include <stdlib.h>

using namespace std;

const int N = 30;

int Empty = -9999;

struct list{

int arr[N];

int last;

};

int insert(list \*l, int x, int p){

if (l->last == Empty) {

l->last = -1;

}

if (l->last >= N){

cout << "list is full" << endl;

}

else{

if (p >= 0 && p < N){

if (p > l->last){

l->last++;

l->arr[l->last] = x;

return l->arr[l->last];

}

else{

for(int i = l->last; i >= p; i --){

l->arr[i + 1] = l->arr[i];

}

l->arr[p] = x;

l->last++;

return l->arr[p];

}

}

else{

cout << "position: " << p <<" out of range " << N << endl;

return Empty;

}

}

}

int del(list \*l, int p){

if (l->last == Empty) {

cout << "list is empty!\n";

return Empty;

}

if (p <= l->last && p >= 0){

int delEL;

delEL = l->arr[p];

for (int i = p; i < l->last; i++){

l->arr[i] = l->arr[i + 1];

}

l->arr[l->last] = Empty;

l->last--;

if (l->last == -1) {

l->last = Empty;

}

return delEL;

}

else{

cout << "position: " << p <<" out of range " << l->last << endl;

return Empty;

}

}

int s\_val(list \*l, int v){

if (l->last != Empty){

for(int i = 0; i <= l->last; i++){

if (l->arr[i] == v){

return i;

}

}

cout << "value: " << v <<" not in the list"<< endl;

return Empty;

}

cout << "list is empty" << endl;

return Empty;

}

int s\_p(list \*l, int p){

if (l->last != Empty){

if (p >= 0 && p <= l->last){

return l->arr[p];

}

cout << "position: " << p <<" out of range " << l->last << endl;

return Empty;

}

cout << "list is empty" << endl;

return Empty;

}

int getINT(){

int k;

int i = 0;

do{

if (i > 0) {

cout << "only integer!\n";

}

i++;

cin.clear();

cin.ignore(cin.rdbuf() -> in\_avail());

cin >> k;

}while(cin.fail());

return k;

}

void printall(list \*l) {

if (l->last != Empty) {

for (int i = 0; i <= l->last; i++) {

cout << "---- " << i << " element: " << l->arr[i] << endl;

}

}

else {

cout << "list is empty" << endl;

}

}

void pushelement(list \*l) {

cout << "input element: ";

int el = getINT();

cout << "input position to push element: ";

int pos = getINT();

int res = insert(l, el, pos);

if (res != Empty){

cout << res << " pushed successfuly!" << endl;

}

}

void deleteelement(list \*l) {

cout << "input index to delete: ";

int pos = getINT();

int res = del(l, pos);

if (res != Empty){

cout << res << " deleted successfuly!" << endl;

}

}

void search\_by\_val(list \*l) {

cout << "input element to search: ";

int el = getINT();

int res = s\_val(l, el);

if (res != Empty){

cout << res << " index"<< endl;

}

}

void search\_by\_ind(list \*l) {

cout << "input index of element to search: ";

int el = getINT();

int res = s\_p(l, el);

if (res != Empty) {

cout << res << "value"<< endl;

}

}

int main() {

list \*l = new list;

l->last = Empty;

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

l->arr[i] = Empty;

}

int k;

do {

cout << "choose what to do:\n\t1- print list\n\t2- push element\n\t3- delete element\n\t4- search element by value\n\t5- search element by index\n\t0- exit\n";

k = getINT();

switch (k){

case 1: printall(l);

cout << endl;

break;

case 2: pushelement(l);

cout << endl;

break;

case 3: deleteelement(l);

cout << endl;

break;

case 4:search\_by\_val(l);

cout << endl;

break;

case 5: search\_by\_ind(l);

cout << endl;

break;

default:

break;

}

} while (k != 0);

}