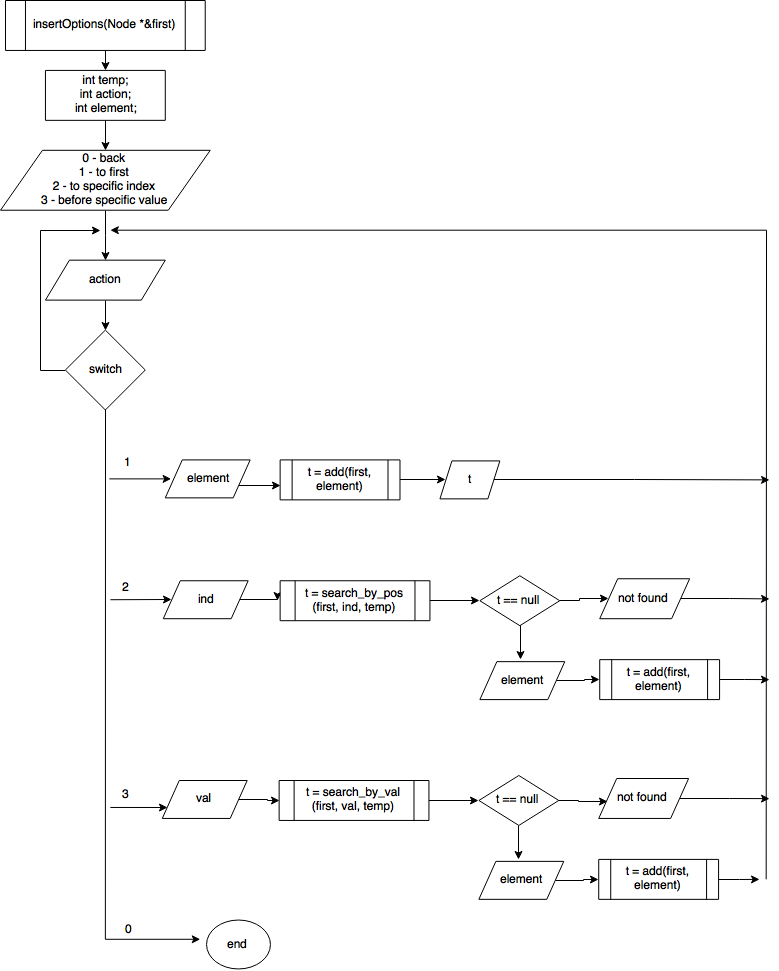
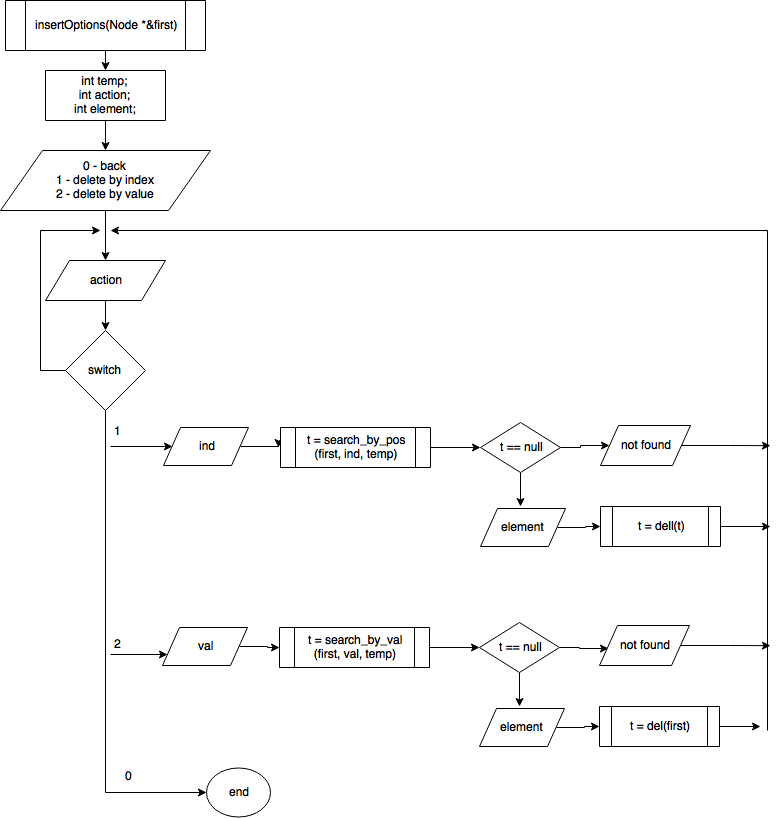
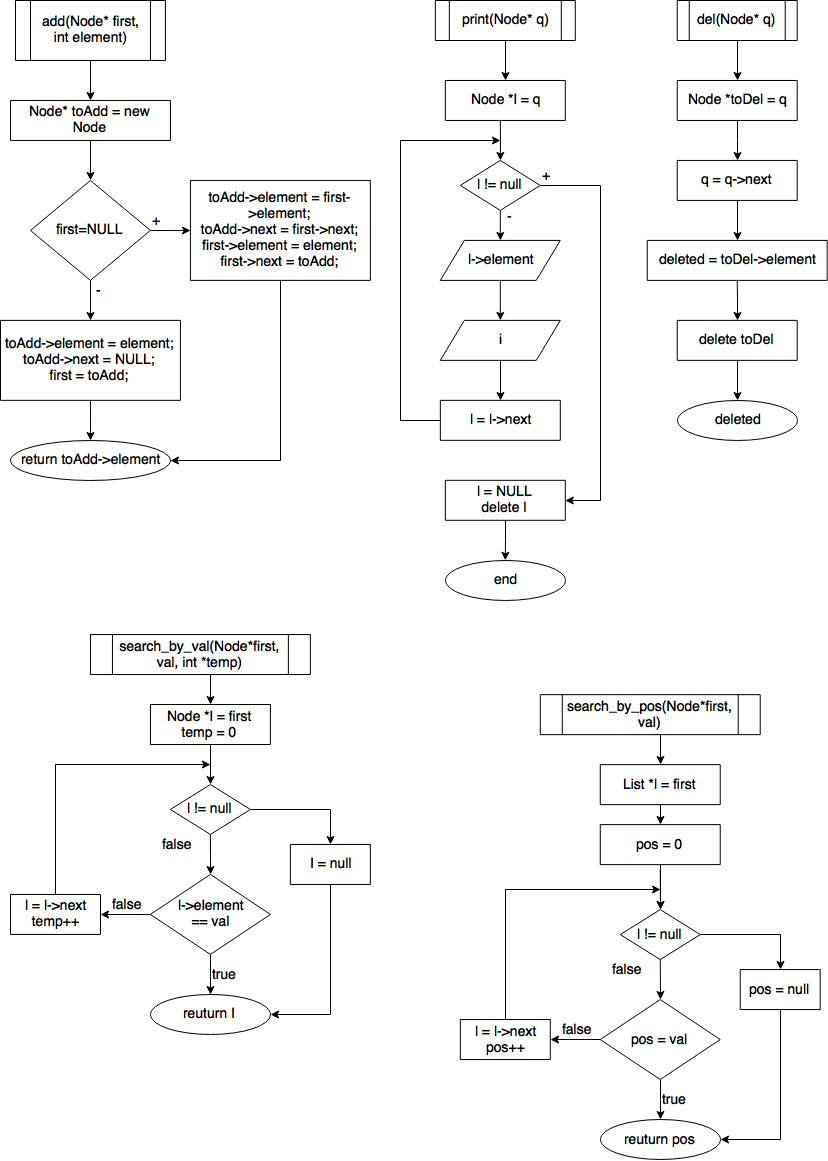
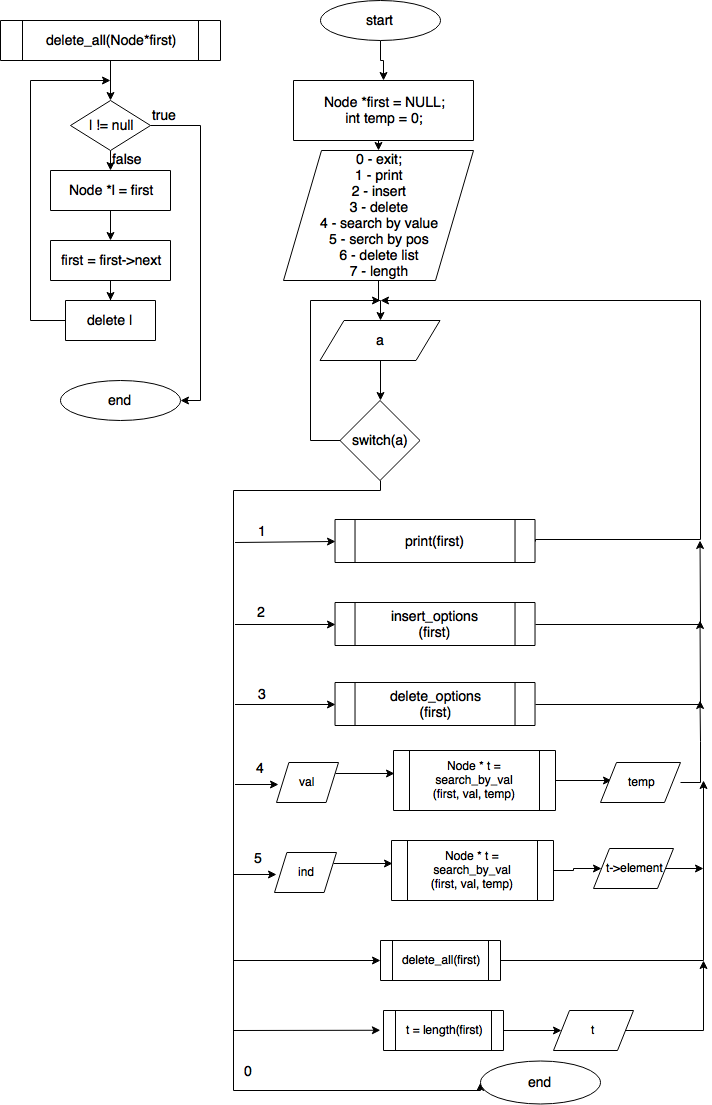
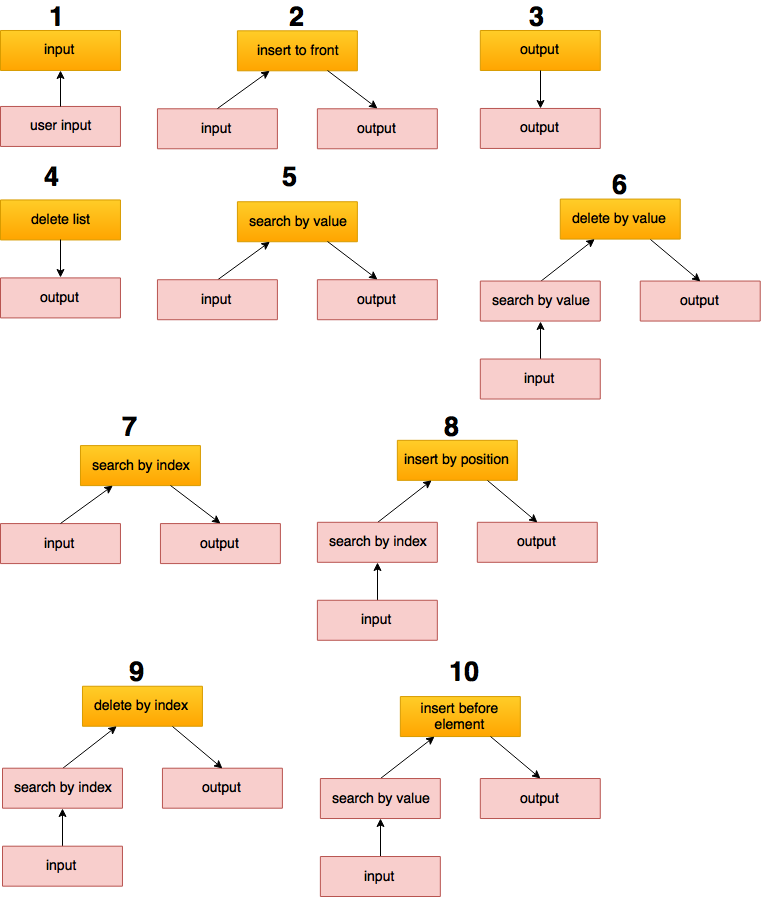
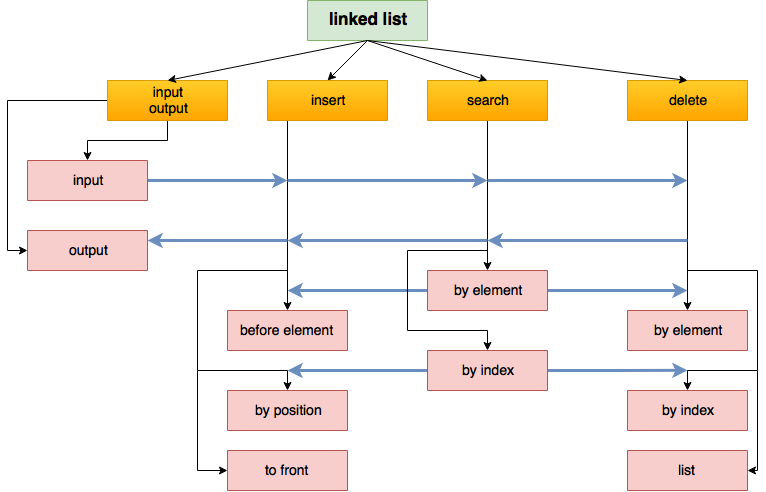
|  |
| --- |
| Kyrgyz State Technical University |
| Algorithms and Data Structures |
| Laboratory work 6 |

|  |
| --- |
| Abakirov Nursultan SE(eng)1-16 |



Design program of one lined list

With functions

* Adding an element to the list to the first position
* Adding an element to the list by position
* Adding an element to the list before specific value
* Deleting an element by position
* Deleting an element by set value
* Search an element by value in the list
* Search an element by position in the list
* Output of the list elements
* Deleting of the list.

**Requirements:**

Type of Input must be an integer.

**Design of diagrams.**

**Design** - The structural method – decomposition.

#include <iostream>

#include <string>

using namespace std;

const string EMPTY = "list is empty\n";

struct Node {

int element;

Node \*next;};

int getINT() {

int k, 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; }

int add(Node \*&first, int element) {

Node \*toAdd = new Node;

if (first == NULL) {

toAdd->element = element;

toAdd->next = NULL;

first = toAdd;

return toAdd->element;}

toAdd->element = first->element;

toAdd->next = first->next;

first->element = element;

first->next = toAdd;

return first->element;}

int del(Node \*&first, Node\*& main) {

Node \*i = main;

if (main != first) {

while (i->next != first) {

i = i->next;}

Node \*toDel = i->next;

i->next = i->next->next;

int deleted = toDel->element;

delete toDel;

return deleted;}

Node \*toDel = first;

first = first->next;

int deleted = toDel->element;

delete toDel;

main = first;

return deleted;}

int delbyval(Node \*&first, int val) {

Node\*found = first;

while (found != NULL) {

if (found->element == val) {

Node \*toDel = found;

found = found->next;

int k = toDel->element;

delete toDel;

return k;}

found = found->next;}}

void print(Node \*first) {

Node \*i = first;

cout << "\nqueue: ";

while (i != NULL) {

cout << i->element << " ";

i = i->next;}

i = NULL;

delete i;

cout << endl;}

Node\* search\_val(Node \*&first, int element, int &temp) {

Node\*found = first;

temp = 0;

while (found != NULL) {

if (found->element == element) {

return found;}

temp++;

found = found->next;}

found = NULL;

return found;}

Node \*search\_pos(Node \*&first, int element, int &temp) {

Node\* found = first;

temp = 0;

while (found != NULL) {

if (temp == element) {

return found;}

temp++;

found = found->next;}

found = NULL;

return found;}

void delAll(Node \*&first) {

while (first != NULL) {

Node\* del = first;

first = first->next;

delete del;}}

void insertOptions(Node \*&first){

int temp;

int action;

int element;

do{

cout << "\nINSERT\n";

cout << "\t0 - back\n";

cout << "\t1 - to back\n";

cout << "\t2 - to specific index\n";

cout << "\t3 - before specific value\n";

action = getINT();

if(action == 0){

return;}

else if(action == 1){

cout << "enter a element: ";

element = getINT();

int inserted = add(first, element);

cout << "inserted: " << inserted << endl;}

else if(action == 2){

if(first != NULL){

cout << "enter index: ";

int ind = getINT();

if(ind >= 0){

Node \*found = search\_pos(first, ind, temp);

if (found != NULL){

cout << "enter a element: ";

element = getINT();

int inserted = add(found, element);

cout << "inserted: " << inserted << endl;}

else{

cout << "index out of range\n";}}

else{

cout << "index out of range\n";}}

else{

cout << "list is empty, element will be inserted to 0 index \n";

cout << "enter a element: ";

element = getINT();

int inserted = add(first, element);

cout << "inserted: " << inserted << endl;}}

else if(action == 3){

if(first != NULL){

cout << "enter specific value: ";

int ind = getINT();

Node \*found = search\_val(first, ind, temp);

if (found != NULL){

cout << "enter a element: ";

element = getINT();

int inserted = add(found, element);

cout << "inserted: " << inserted << endl;}

else{

cout << "value not found\n";}}

else{

cout << "list is empty, element will be inserted to 0 index \n";

cout << "enter a element: ";

element = getINT();

int inserted = add(first, element);

cout << "inserted: " << inserted << endl;}}

}while(action != 0);}

void deleteOptions(Node \*&first){

int action;

int temp;

do{

cout << "\t0 - back\n";

cout << "\t1 - delete by value\n";

cout << "\t2 - delete by index\n";

action = getINT();

if (action == 0){return;}

else if (action == 9) {

int t = del(first, first);

cout << t << endl;}

else if(action == 1){

if (first == NULL) {

cout << EMPTY;

break;}

cout << "element: ";

int element = getINT();

Node \*f\_value = search\_val(first, element, temp);

if (f\_value == NULL) {

cout << "value in list not found\n";

break;}

else {

int t = del(f\_value, first);

cout << "deleted element: " << t << endl;}}

else if(action == 2){

if (first == NULL) {

cout << EMPTY;

break;}

cout << "index: ";

int element = getINT();

if (element >= 0) {

Node \*f\_value = search\_pos(first, element, temp);

if (f\_value == NULL) {

cout << "value in list not found\n";

break;}

else {

int t = del(f\_value, first);

cout << "deleted element: " << t << endl;}}

else {

cout << "position must be positive " << endl;}}

}while(action != 0);}

int main() {

Node \*first = NULL;

int temp = 0;

int action;

do {

cout << "\t0 - exit\n";

cout << "\t1 - print\n";

cout << "\t2 - insert..\n";

cout << "\t3 - delete..\n";

cout << "\t4 - search by value\n";

cout << "\t5 - search by position\n";

cout << "\t6 - delete list\n";

cout << "\t7 - length of list\n";

action = getINT();

if (action == 0) {

break;}

else if (action == 1) {

if (first == NULL) {

cout << EMPTY;}

else {

print(first);}}

else if (action == 2) {

insertOptions(first);}

else if(action == 3){

deleteOptions(first);}

else if (action == 4) {

if (first == NULL) {

cout << EMPTY;}

else {

int element = getINT();

Node\* found = search\_val(first, element, temp);

if (found == NULL) {

cout << "element not found\n";}

else {

cout << "found element: " << found->element << " index: " << temp << endl;}

temp = 0;}}

else if (action == 5) {

if (first == NULL) {

cout << EMPTY;}

else {

int element = getINT();

if (element >= 0) {

Node\*found = search\_pos(first, element, temp);

if (found == NULL) {

cout << "element not found\n";}

else {

cout << "found element: " << found->element << " index: " << temp << endl;}

temp = 0;}

else {

cout << "position must be positive " << endl;}}}

else if (action == 6) {

if (first != NULL) {

delAll(first);}

cout << EMPTY;}

else if (action == 7) {

Node \*pv = first;

int count = 0;

while (pv != NULL) {

count++;

pv = pv->next;}

cout << "length: " << count << endl;

pv = NULL;

delete pv;}

else if (action == 9) {

int a = del(first, first);}

} while (action != 0);

return 0;}