ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | М. А. Мурашова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «ЛИНЕЙНЫЕ И ЦИКЛИЧЕСКИЕ СПИСКИ» |
| по курсу: СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

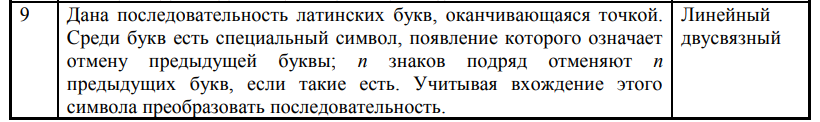
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134К |  |  |  | Иванов И.В. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**1) Цель работы:** Целью работы является изучение структур данных «линейный список» и «циклический список», а также получение практических навыков их реализации.

**2) Задание:**

ВАРИАНТ 9  
Реализовать структуры данных «линейный список» и «циклический список» в соответствии с заданным вариантом. Дополнительно программа должна осуществлять следующие операции:   
1) Добавление/удаление элемента в список (с клавиатуры);   
2) Вывод исходного и результирующего списков на экран;   
3) Если списки являются многочленами, в выводе должна быть отражена степень каждого элемента.



**3) ЛИСТИНГ ПРОГРАММЫ:**

#include <iostream>

#include <string>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

using namespace std;

int flag = 0; // the menu option

int k1 = 0; // equals 1 if the dot is between symbols

char end1[] = ".";

char dell[] = "\0";

char space[] = " ";

struct node {

char key;

node\* prev;

node\* next;

};

node\* head = NULL;

node\* first = NULL;

node \*temp = NULL;

node\* tail = NULL;

void traverse()

{

node\* ptr = head; // nodes points towards head node

while (ptr != NULL) {

int d = 0;

d = ptr->key;

printf("%c ", (int)(d));

ptr = ptr->next;

}

printf("\n");

}

void addnode(char k)

{

node\* ptr = (node\*)malloc(sizeof(node));

ptr->key = k; //key to value k

ptr->next = NULL;

ptr->prev = NULL;

if (head == NULL) {

head = ptr;

first = head;

tail = head;

}

else { // add at the end

temp = ptr;

first->next = temp;

temp->prev = first;

first = temp;

tail = temp;

}

traverse();

cout << endl;

}

void insertatend(char k)

{

node\* ptr = (node\*)malloc(sizeof(node));

ptr->key = k; //Key to value k

ptr->next = NULL;

ptr->prev = NULL;

if (head == NULL) {

first = ptr;

first = head;

tail = head;

}

else { // add at the end

temp = ptr;

temp->prev = tail;

tail->next = temp;

tail = temp;

}

traverse();

cout << endl;

}

void delatend()

{

// Mode tail to the prev

tail = tail->prev;

tail->next = NULL;

traverse();

cout << endl;

}

void cinstr(char a[]) {

cout << "enter your str: ";

cin >> a;

cout << "\n";

}

void outt(char a[]) {

cout << "original input: " << a << endl;

}

int menudu() {

cout << "'\nMENU\n 1.enter str\n 2.cout the result\n choose the thing : ";

cin >> flag;

return flag;

}

void menu2(char a[], int k1) {

menudu();

if (flag == 1) {

cinstr(a);

//for (int q = 0; q < 1; q++) {

cout << "add the node&" << endl;

addnode(\*space);

for (int q = 0; q < strlen(a); q++) {

cout << "add at the end@" << endl;

if (q == strlen(a) - 1 && a[q] != \*end1) {

cout << "you need a dot at the end of your string" << endl;

exit(0);

}

if (a[q] == \*end1 && q != strlen(a) - 1) { // if dot and not end - error

k1 = 1;

cout << " '.' should be the end. enter again\n";

break;

}

if (k1 != 1 && a[q] != \*dell) { // if dot the end and [q] not < - add at end

insertatend(a[q]);

}

else if (q != 0 && k1 != 1 && a[q] == \*dell) { // if dot the end and [q] < - del at [q]

insertatend(a[q]);

cout << "deleting < : " << endl;

delatend();

cout << "deleting letter : " << endl;

delatend();

}

else {

cout << "what the situation??" << endl;

}

}

menudu();

while (flag != 2) {

cinstr(a);

for (int q = 0; q < 1; q++) {

cout << "add the node&" << endl;

addnode(a[q]);

}

for (int q = 1; q < strlen(a); q++) {

cout << "add to end@" << endl;

if (q == strlen(a) - 1 && a[q] != \*end1) {

cout << "wtf u need a dot" << endl;

exit(0);

}

if (a[q] == \*end1 && q != strlen(a) - 1) { // if dot and not end - error

k1 = 1;

cout << " '.' should be the end. enter again\n";

break;

}

if (k1 != 1 && a[q] != \*dell) { // if dot the end and [q] not < - add at end

insertatend(a[q]);

}

else if (q != 0 && k1 != 1 && a[q] == \*dell) { // if dot the end and [q] < - del at [q]

insertatend(a[q]);

cout << "deleting < : " << endl;

delatend();

cout << "deleting letter : " << endl;

delatend();

}

else {

cout << "what the situation??" << endl;

}

}

menudu();

}

if (flag == 2) {

cout << "\n";

outt(a);

}

}

else if (flag > 2) {

cout << "no such option.";

}

else if (flag == 2) {

cout << "cant do 2 before input.";

}

else {

cout << "error option.";

}

}

int main()

{

char a[100] = { '\0' };

cout << "choose the deleting symbole: ";

cin >> dell;

menu2(a, k1);

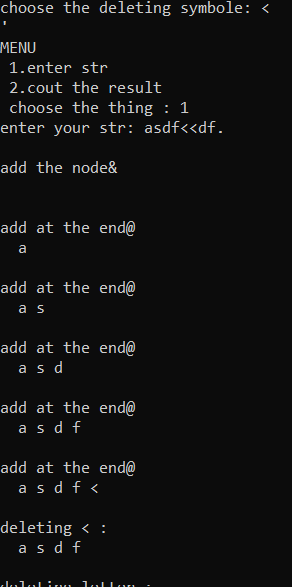
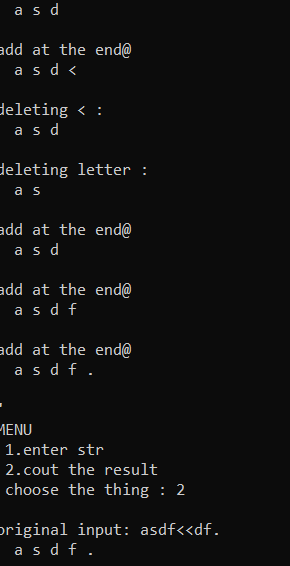
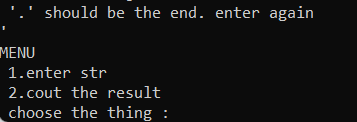
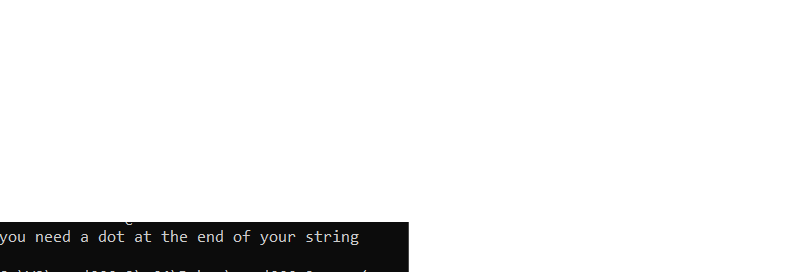
traverse();

return 0;

}

**4)** **Контрольные примеры:**

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| asdf<<df. | a s d f . |
| asaad.sada | '.' should be the end. enter again |
| a<sdf< | you need a dot at the end of your string |

1. ****
2. 
3. 

**5) Выводы:**

Был разработан алгоритм, который осуществляет создание узла с указателями предыдущий и следующий равных нулю, и последующее добавление элементов в конец списка, одновременно с этим меняющие указатели первого узла следующий на предыдущий второго узла и тд с остальными. При удалении же осуществляется смена указателя последнего(i) узла предыдущий на следующий (i-2) узла. Таким образом мы меняем зависимости