Пермский Национальный Исследовательский Политехнический Университет

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Лабораторная работа № 11.1

по дисциплине

«Основы алгоритмизации и программирования»

Тема: «Динамические структуры данных. Однонаправленный список»

Вариант № 17

Выполнил:

студент группы РИС-20-2б

Пономарев Егор Витальевич

Проверила:

доцент кафедры ИТАС

Полякова О. А.

Пермь, 2021

**Цель работы**

Получить практические навыки работы с однонаправленными списками.

**Постановка задачи**

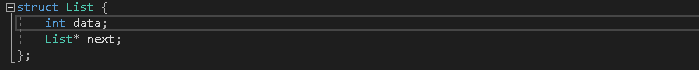
* Сформировать однонаправленный список.
* Распечатать полученную структуру.
* Выполнить обработку структуры в соответствии с заданием.
* Распечатать полученный результат.
* Удалить соответствующую структуру из памяти.

Задание варианта:

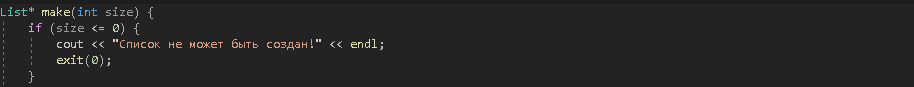
1. Тип информационного поля int.
2. Удалить из списка все элементы с четными информационными полями.

**Анализ задачи**

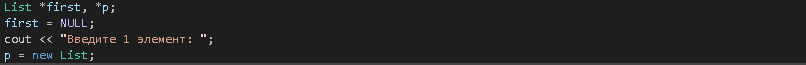
1. Для решения задачи необходимо:
2. Необходимо описать структуру **List** с информационным полем **int** и указателем на следующий элемент **List\* next**.



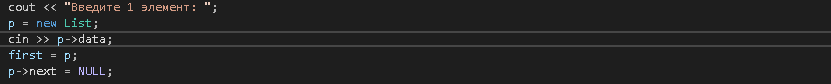
1. Вводим размер списка и передаем его в функцию создания списка **List\*make (int size).**
2. Проверяем возможно ли создать список (если size = 0, выходим из программы).



1. Объявляем указатели **first** и **p** типа **List** и приравниваем **first** к NULL. (Первый элемент пока что указывает на NULL, то есть он пустой).



1. Вводим элемент в **p**, динамически выделив для него память. Приравниваем **first** к **p**. Таким образом, p теперь указывает на **NULL**.



1. С помощью цикла вводим остальные элементы. Для этого объявляем временную переменную **List\* h = new List.** Не забываем работать с указателями.



1. Возвращаем указатель **first**.
2. В функции печати с помощью цикла печатаем все элементы, пока указатель не равен NULL, постепенно переходя от первого элемента к последнему.



1. В функции удаления учтем случай, если в списке всего 1 элемент и тот четный, объявив флажок. В таком случае просто удаляем его с помощью delete и флажок перекидываем на 0.



1. Если же элементов больше, чем 1, проходим по всем элементам и проверяем на четность. Если число четное, сохраняем его во временную переменную (чтобы потом удалить мусор и не потерять его). Указатель предыдущего элемента направляем на следующий, а временную переменную удаляем.



1. В ходе работы были использованы следующие типы данных:
2. Int для счетчиков и размера списка.
3. Структура данных List с информационным полем Int и указателем List\*

3. Ввод и вывод осуществляется с помощью cin и cout/

4. В работе использовались циклы while, for и условные ветвления If.

**Блок-схема**

**Код С++**

#include <iostream>

#include <locale.h>

#include <cstdlib>

using namespace std;

struct List {

int data;

List\* next;

};

List\* make(int size) {

if (size <= 0) {

cout << "Список не может быть создан!" << endl;

exit(0);

}

List \*first, \*p;

first = NULL;

cout << "Введите 1 элемент: ";

p = new List;

cin >> p->data;

first = p;

p->next = NULL;

for (int i = 2; i <= size; i++) {

List \*h = new List;

p->next = h;

p = p->next;

cout << "Введите " << i << " элемент: ";

cin >> p->data;

p->next = NULL;

}

return first;

}

void print(List \*first) {

if (first == NULL)

cout << "Список пуст\n";

else {

List \*p = first;

while (p != NULL) {

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

}

List\* del(List \*first) {

bool f = 1;

List \*p = first;

while (f && ((p->data) % 2 == 0)) {

List \*q = first;

if (p->next != NULL)

first = p->next;

else {

first = NULL;

f = 0;

}

delete q;

p = first;

}

while (f && (p->next != NULL)) {

if ((p->next->data) % 2 == 0) {

List \*q = p->next;

p->next = p->next->next;

delete q;

}

else p = p->next;

}

return first;

}

int main()

{

setlocale(LC\_ALL, "Rus");

int size;

cout << "Введите количество элементов списка: ";

cin >> size;

List \*list = make(size);

print(list);

List \*list1 = del(list);

print(list1);

return 0;

}

**Результаты выполнения**

