|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Практическая работа № 1 | | |
| по дисциплине «Информационные технологии и основы программирования» | | |
| **структура данных: список** | | |
|  | | |
|  | Бригада №3 | Ерощенко Артем, Кауфман Яна, Ковалева МАрия |
| Группа ПМИ-33 |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | Дворецкая виктория константиновна |
|  |  |
| Новосибирск,2024 | | |

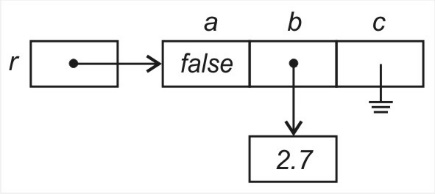
**1.** **Цель работы:** получить практические навыки в организации и обработке списков на языке СИ.

**2. Задание:** Для указанного ниже представления списка, опишите на Си

соответствующий тип *список*, определите переменную *r* (и, если надо,

вспомогательные переменные) и напишите последовательность

операторов для создания этого списка:



**3. Анализ задачи:**

**Входные данные:** переменные типа bool и float\*

**Выходные данные:** элементы списка

**Метод решения:**

Программа реализует односвязный список. Возможно удаление, добавление и выгрузка элементов списка.

**4. Текст программы:**

#include <iostream>

namespace List{

struct Node{

bool fst;

float \*sec;

Node \*next;

};

char insert\_node(List::Node \*head, int ind, bool fst, float \*sec){ // Insert node to ind

List::Node \*buff\_node = head;

for(int i = 0; i < ind - 1; i++){ // Iterate list to Node before ind

if(buff\_node->next != NULL){

buff\_node = buff\_node->next;

} else{

return 0;

}

}

List::Node \*new\_node = new List::Node(); // Create a new node and fill it with data

new\_node->fst = fst;

new\_node->sec = sec;

new\_node->next = buff\_node->next; // Change pointers

buff\_node->next = new\_node;

return 1;

}

char del\_node(List::Node \*head, int ind){ // Delete ind

List::Node \*buff\_node = head;

for(int i = 0; i < ind - 1; i++){ // Iterate list to Node before ind

if(buff\_node->next != NULL){

buff\_node = buff\_node->next;

} else{

return 0;

}

}

List::Node \*deleted = buff\_node->next;

if(buff\_node->next != NULL){

buff\_node->next = buff\_node->next->next; // Change pointers

} else{

return 0;

}

delete deleted; // Free deleted node

return 1;

}

char free\_all(List::Node \*head){ // Free all nodes

List::Node \*buff\_node = head, \*prev\_buff\_node = NULL;

while(buff\_node != NULL){

prev\_buff\_node = buff\_node;

buff\_node = buff\_node->next;

delete prev\_buff\_node;

}

return 1;

}

void print\_list(List::Node \*head){ // Print the whole list

while(head != NULL){

std::cout << "bool: " << head->fst << "; float: " << \*(head->sec) << "; -> ";

head = head->next;

}

std::cout << '\n';

}

}

int main(){

float num1 = 2.7, num2 = 3.3;

List::Node \*p = new List::Node();

p->fst = false;

p->sec = &num1;

p->next = NULL;

List::insert\_node(p, 1, true, &num1);

List::insert\_node(p, 2, false, &num2);

List::print\_list(p);

List::del\_node(p, 1);

List::print\_list(p);

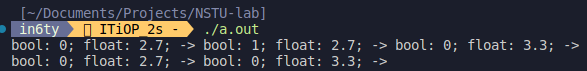
List::free\_all(p);

return 0;

}

**5. Тестирование программы:**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Выходные данные | Примечание |
| 1 |  |  |  |

****