

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  **информационных**  **технологий** | **Кафедра**  **информационных технологий и вычислительных систем** |

**КУРСОВАЯ РАБОТА**  
ПО ДИСЦИПЛИНЕ

«СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| СТУДЕНТА | *2* | КУРСА | | *бакалавриата* | ГРУППЫ | *ИДБ-20-01* |
|  | | | *(уровень профессионального образования)* | |  | |

|  |
| --- |
| **Кудряшова Дмитрия Вадимовича** |
| *(ФИО)* |

ТЕМА РАБОТЫ

|  |
| --- |
| МНОЖЕСТВО ПРЕДЛОЖЕНИЙ |

|  |  |
| --- | --- |
| Направление: | 09.03.01 Информатика и вычислительная техника |
| Профиль подготовки: | Программное обеспечение средств вычислительной техники и автоматизированных систем |

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет сдан «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г. | | | |
|  |  |  |  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
|  | | | |
| Преподаватель | Лакунина О.Н., ст. преподаватель |  |  |
|  | *(Ф.И.О., должность, степень, звание.)* |  | *(подпись)* |

МОСКВА 2021

**Оглавление**

Задание на курсовую работу………………………………………………2

Описание структур данных……………………………………………….3

Конечная схема структуры данных ………………………………………5

Описание структур на языке Си…………………………………………..6

Схема вызова функций…………………………………………………….7

Список функций и их значение……………………………………………8

Исходный код программы с комментариями…………………………….9

# Задание на курсовую работу

Написать программу, реализующую логическую структуру данных – множество предложений.

Программа должна работать в диалоговом режиме.

Каждая операция должна быть реализована в виде отдельной функции.

Множество должно быть реализовано на базе структуры хранения – двусвязный список.

Предложение должно быть реализовано на базе структуры хранения – односвязный список.

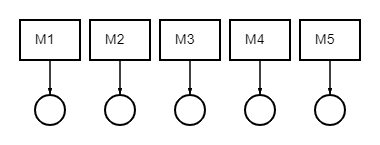
Написать отчет по курсовой работе.

**Описание структур данных**

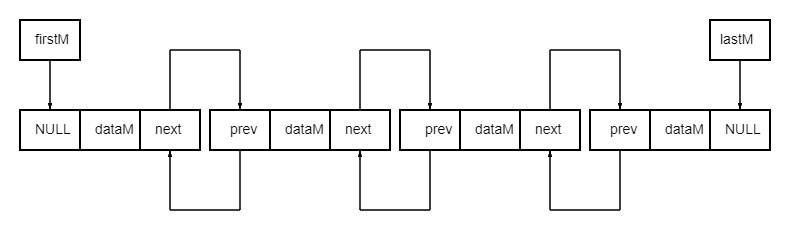
**Множество**

Множество – логическая структура данных, в которой элементы хранятся неупорядоченно, где нет повторяющихся элементов.

Логическая структура:

****

Физическая структура (двусвязный список):



|  |  |
| --- | --- |
| firstM | Указатель на начало |
| lastM | Указатель на конец |
| prev | Указатель на предыдущий элемент |
| dataM | Элемент данных |
| next | Указатель на следующий элемент |

Список реализуемых функций:

1.Начать работу

2.Сделать множество пустым

3.Проверить множество на пустоту

4.Удалить выбранный элемент из множества

5.Взять какой-либо элемент из множества

6.Добавить элемент в множество

7.Проверить элемент на принадлежность множеству

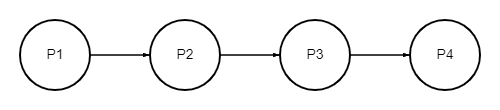
8.Вывести множество

9.Закончить работу

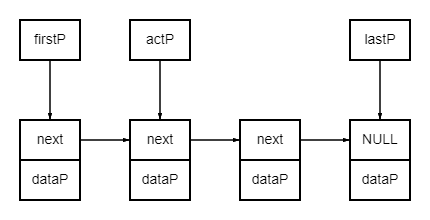
**Предложение**

Предложение – структура данных, представляющая из себя односвязный список, где каждый элемент содержит ссылку на следующий элемент.

Логическая структура:



Физическая структура (односвязный список):



|  |  |
| --- | --- |
| firstP | Указатель на начало |
| lastP | Указатель на конец |
| actP | Рабочий указатель |
| dataP | Элемент данных |
| next | Указатель на следующий элемент |

Список реализуемых функций:

1.Начать работу

2.Сделать предложение пустым

3.Проверить предложение на пустоту

4.Установить рабочий указатель в начало предложения

5.Проверить, находится ли указатель в конце предложения

6.Передвинуть рабочий указатель вперёд

7.Вывести элемент за указателем

8.Удалить элемент за указателем

9.Взять слово за указателем

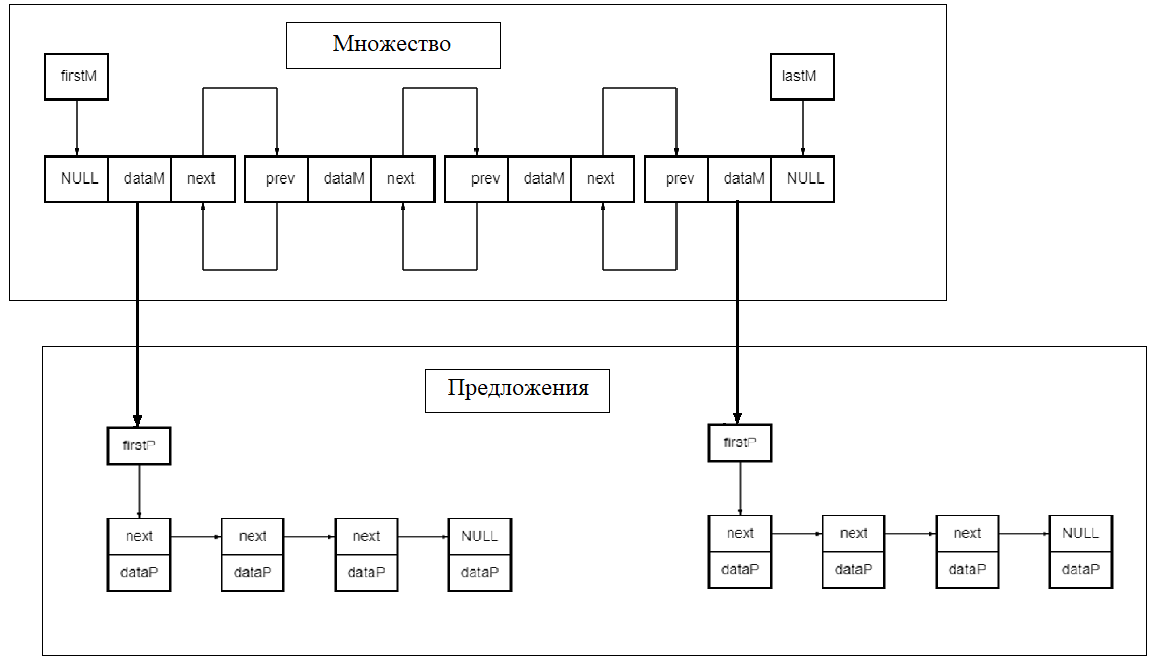
10.Изменить слово за указателем;

11.Добавить слово за указателем

12.Вывести предложение

13.Закончить работу

**Конечная схема структуры данных**

****

**Описание структур на языке Си**

struct mnoz { //Узел множества

predl\* data; //Элемент данных множества - предложение

struct mnoz\* next; //Указатель на следующий элемент множества

struct mnoz\* prev; //Указатель на предыдущий элемент множества

};

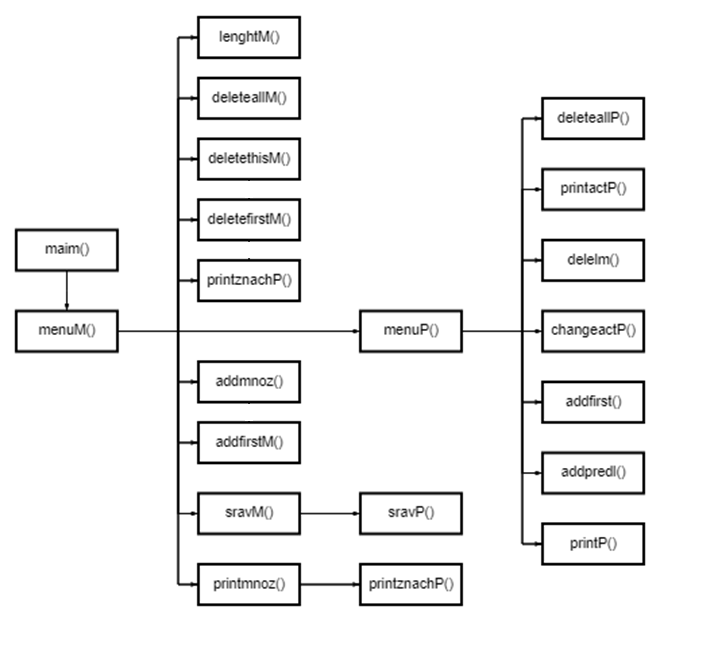
struct predl { //Узел предложения

string\* data; //Элемент данных предложения

predl\* next; //Указатель на следующий элемент

};

**Схема вызова функций**

****

**Список функций и их значение**

**Множество:**

**menuM() - меню множества**

**lenghtM() - Подсчёт количества элементов множества**

**deleteallM() - Удаление всех элементов множества**

**deletethisM() - Удаление выбранного элемента множества при его наличии**

**deletefirstM() - Удаление первого элемента множества**

**addmnoz() - Создание второго и последующих элементов множества**

**addfirstM() - Добавление первого элемента множества**

**sravM() - Сравнение элементов множества с предложением**

**sravP() - Сравнение двух предложений**

**printmnoz() - Вывод множества на экран**

**printznachP() – Вывод элемента данных множества на экран**

**Предложение:**

**menuP() – меню предложения**

**deleteallP() - Удаление всех элементов предложения**

**printactP() - Вывод элемента предложения за указателем**

**delelm() - Удаление элемента за указателем**

**changeactP() - Изменение значения элемента за указателем**

**addfirst() - Создание первого элемента первого элемента предложения**

**addpredl() - Создание элемента предложения за указателем**

**printP() - Вывод элементов предложения на экран в строчку целиком**

**Исходный код программы с комментариями**

#define \_SRT\_SECURE\_NO\_WARNINGS

#include <string>//Работа со строками

#include <iostream>//Ввод-вывод

#include <cstdlib>//Выделение памяти

using namespace std;

struct predl {

string\* data;

predl\* next;

};//Создание структуры предложения//Создание структуры предложения

struct mnoz {

predl\* data;

struct mnoz\* next;

struct mnoz\* prev;

}; //Создание структуры множества

predl\* addfirst(string data)

{

predl\* P1 = (predl\*)calloc(1, sizeof(predl));

string\* f = new string();

\*f = data;

if (P1 == nullptr) { std::cout << "Ошибка. Память переполненна.\n"; exit(1); }

P1->next = nullptr;

P1->data = f;

return P1;

}//Создание первого элемента первого элемента предложения, функция возвращает его адрес

predl\* addpredl(predl\* aP, string data)

{

predl\* addP, \* p;

addP = (predl\*)calloc(1, sizeof(predl));

if (addP == nullptr) { std::cout << "Ошибка. Память переполненна.\n"; exit(1); }

p = aP->next;

aP->next = addP;

addP->next = p;

string\* f = new string();

\*f = data;

addP->data = f;

return addP;

}//Создание элемента предложения за указателем

void printactP(predl\* fP, predl\* aP)

{

predl\* f = fP;

int i = 1;

while (f != NULL) {

if (f == aP)

cout << "Элемент за указателем: " << "\t" << \*(f->data) << "\n";

f = f->next;

i++;

}

printf("\n");

} //Вывод элемента предложения за указателем

void printP(predl\* fP, predl\* aP)

{

predl\* f = fP;

int i = 1;

cout << "Предложение:\n";

while (f != NULL) {

if (f == aP)

cout << "Элемент\* " << i << "\t" << \*(f->data) << "\n";

else

cout << "Элемент " << i << "\t" << \*(f->data) << "\n";

f = f->next;

i++;

}

printf("\n");

} //Вывод элементов предложения на экран в строчку целиком

predl\* changeactP(predl\* fP, predl\* aP, string data)

{

predl\* change = fP;

while (!(change == aP))

change = change->next;

string\* f = new string();

\*f = data;

change->data = f;

return fP;

} //Изменение значения элемента за указателем

void deleteallP(predl\* uk)

{

predl\* a = uk;

predl\* buffer = uk;

do {

buffer = a->next;

free(a);

a = buffer;

} while (a != NULL);

} //Удаление всех элементов предложения

predl\* delelm(predl\* aP, predl\* fP)

{

predl\* dell;

dell = fP;

while (dell->next != aP)

dell = dell->next;

dell->next = aP->next;

free(aP);

return dell;

}//Удаление элемента за указателем

predl\* menuP()

{

string data;

predl\* firstP = NULL;

predl\* lastP = NULL;

predl\* actP = NULL;

bool begin = false;

int choice = -1;

while (1)

{

cout << "\nСтруктура данных: предложение\n";

cout << "Доступные операции:\n";

cout << "1\tНачать работу\n";

cout << "2\tСделать предложение пустым\n";

cout << "3\tПроверить предложение на пустоту\n";

cout << "4\tУстановить рабочий указатель в начало предложения\n";

cout << "5\tПроверить, находится ли указатель в конце предложения\n";

cout << "6\tПередвинуть рабочий указатель вперёд\n";

cout << "7\tВывести слово за указателем\n";

cout << "8\tУдалить слово за указателем\n";

cout << "9\tВзять слово за указателем\n";

cout << "10\tИзменить слово за указателем\n";

cout << "11\tДобавить слово за указателем\n";

cout << "12\tВывести предложение\n";

cout << "13\tЗакончить работу с предложением\n";

cin >> choice;

system("cls");

if ((choice > 13) || (choice < 1))

cout << "Введите число от 1 до 13\n\n";

else if

((begin == false) && (choice != 1))

cout << "Необходимо начать работу. Пункт 1.\n\n";

else

switch (choice)

{

case 1:

cout << "Работа начата\n\n";

begin = true;

break;

case 2:

{

if (firstP == NULL)

{

cout << "Предложение уже пусто\n";

break;

}

deleteallP(firstP);

actP = lastP = firstP = NULL;

cout << "Предложение очищено";

break;

}

case 3:

{

if (firstP == NULL)

cout << "Предложение пусто\n";

else

cout << "Предложение НЕ пусто\n";

break;

}

case 4:

{

if (firstP == NULL)

{

cout << "Предложение пусто\n";

break;

}

actP = firstP;

cout << " Рабочий указатель установлен в начало предложения\n";

break;

}

case 5:

{

if (firstP == NULL)

{

cout << "Предложение пусто\n";

break;

}

if ((actP->next) == nullptr)

cout << "Указатель находится в конце предложения\n";

else

cout << "Указатель находится НЕ в конце предложения\n";

break;

}

case 6:

{

if (firstP == NULL)

{

cout << "Предложение пусто\n";

break;

}

if (actP->next) {

actP = actP->next;

cout << "Указатель передвинут вперёд\n";

}

else

cout << "Указатель находится в конце предложения\n";

break;

}

case 7:

{

if (firstP == NULL)

{

cout << "Предложение пусто\n";

break;

}

if (actP->next) printactP(firstP, actP->next); else cout << "За указателем нет элемента\n";

break;

}

case 8:

{

if (firstP == NULL)

{

printf("Предложение пусто\n");

break;

}

if (actP->next) actP = delelm(actP->next, firstP); else cout << "За указателем нет элемента\n";

break;

}

case 9:

{

if (firstP == NULL)

{

printf("Предложение пусто\n");

break;

}

if (actP->next) {

cout << "Взятое слово: " << \*(actP->next->data) << "\n";

actP = delelm(actP->next, firstP);

}

else cout << "За указателем нет элемента\n";

break;

}

case 10:

{

if (firstP == NULL)

{

printf("Предложение пусто\n");

break;

}

if (actP->next) {

cout << "Введите строку\n";

cin >> data;

firstP = changeactP(firstP, actP->next, data);

}

else cout << "За указателем нет элемента\n";

break;

}

case 11:

{ cout << "Введите строку\n";

cin >> data;

if (firstP == NULL)

{

lastP = actP = firstP = addfirst(data);

}

else

{

actP->next = addpredl(actP, data);

if (actP->next == NULL) lastP = actP;

}

break;

}

case 12:

{ printP(firstP, actP); break; }

case 13:

return firstP;

}

if(firstP) printP(firstP, actP);

}

}//Меню для работы с предложением

int sravP(predl\* a, predl\* b)

{

while (a != NULL) {

if (\*(a->data) != \*(b->data)) return 0;

a = a->next;

b = b->next;

}

return 1;

} //Сравнение двух предложений

void printznachP(predl\* uk)

{

predl\* a = uk;

while (a != NULL) {

cout << \*(a->data) << "\t";

a = a->next;

}

printf("\n");

} //Вывод элемента данных множества на экран

mnoz\* addfirstM(predl\* data)

{

mnoz\* a;

a = (mnoz\*)malloc(sizeof(mnoz));

if (a == NULL) { std::cout << "Ошибка - память переполненна\n"; exit(1); }

else {

a->data = data;

a->next = NULL;

a->prev = NULL;

}

return a;

} //Добавление первого элемента множества, функция возвращает его адрес

mnoz\* addmnoz(mnoz\* a, predl\* d)

{

while (a->next != NULL) {

a = a->next;

}

mnoz\* addedmnoz = (mnoz\*)malloc(sizeof(mnoz));

if (addedmnoz == NULL) { std::cout << "Ошибка - память переполненна\n"; exit(1); }

else {

addedmnoz->next = NULL;

addedmnoz->data = d;

addedmnoz->prev = a;

a->next = addedmnoz;

}

return a->next;

} //Создание второго и последующих элементов множества

void printmnoz(mnoz\* a, mnoz\* b)

{

int i = 1;

printf("Множество:\n");

while (a != NULL) {

printf("Элемент[%d] ", i);

printznachP(a->data);

a = a->next;

i++;

}

printf("\n");

} //Вывод множества на экран

int lenghtM(mnoz\* uk)

{

int len = 0;

if (uk == NULL) return 0;

else {

while (uk != NULL) {

uk = uk->next;

len++;

}

return len;

}

} //Подсчёт количества элементов множества

void deleteallM(mnoz\* uk)

{

mnoz\* a = uk;

mnoz\* buffer = uk;

do {

buffer = a->next;

free(a);

a = buffer;

} while (a != NULL);

} //Удаление всех элементов множества

int sravM(mnoz\* a, predl\* b)

{

while (a != NULL) {

if (sravP(a->data, b)) return 1;

a = a->next;

}

return 0;

} //Сравнение элементов множества с предложением

mnoz\* deletefirstM(mnoz\* e)

{

if (!(e->next)) return NULL;

mnoz\* buffer = e->next;

free(e);

e = buffer;

if (!(buffer))

e->prev = NULL;

return e;

} //Удаление первого элемента множества

mnoz\* deletethisM(mnoz\* e, mnoz\* last, predl\* b)

{

mnoz\* head = e;

if (sravP(e->data, b)) {

cout << "Выбранное предложение удалено из множества\n";

return deletefirstM(e);

}

if (sravP(last->data, b)) {

last->prev->next = NULL;

free(last);

cout << "Выбранное предложение удалено из множества\n";

return e;

}

while (e != NULL) {

if (sravP(e->data, b)) {

mnoz\* prev = e->prev;

mnoz\* next = e->next;

e->next->prev = prev;

e->prev->next = next;

free(e);

cout << "Выбранное предложение удалено из множества\n";

return head;

}

e = e->next;

}

cout << "Выбранное предложение отсутствует в множестве\n";

return head;

}//Удаление выбранного элемента множества при его наличии

mnoz\* menuM()

{

int number = -1;

int mode = 0;

predl\* takeP = 0;//Указатель на предложение

mnoz\* firstM = NULL; //Указатель на первый элемент множества

mnoz\* lastM = NULL; //Указатель на последний элемент множества

while (1) {

cout << "\nСтруктура данных: множество\n";

cout << "Доступные операции:\n";

cout << "1.\tНачать работу с множеством\n";

cout << "2.\tСделать множество пустым\n";

cout << "3.\tПроверить множество на пустоту\n";

cout << "4.\tУдалить выбранное предложение из множества\n";

cout << "5.\tВзять какое-либо предложение из множества\n";

cout << "6.\tДобавить предложение в множество\n";

cout << "7.\tПроверить предложение на принадлежность множеству\n";

cout << "8.\tВывести множество\n";

cout << "9.\tЗакончить работу с множеством\n\n";

cout << "101.\tЗавершить работу программы\n\n";

cout << "Выберите пункт меню:\n";

cin >> number;

system("cls");

if (number == 101) return firstM;

if (number == 1 && mode == 1)

cout << "Вы уже начали работу с множеством.\n";

if ((number == 1) && (mode == 0)) {

mode = 1;

cout << "Работа с множеством начата\n";

}

if (mode == 1)

{

if (number == 2)

{

if (lenghtM(firstM)) {

deleteallM(firstM);

lastM = NULL;

firstM = NULL;

cout << "Множество очищено.\n";

}

else cout << "Множество пусто.\n";

}

if (number == 3)

if (lenghtM(firstM))

cout << "Множество НЕ пусто\n";

else

cout << "Множество ПУСТО\n";

if (number == 4)

{

if (lenghtM(firstM)) firstM = deletethisM(firstM, lastM, menuP());

else cout << "Множество пусто.\n";

}

if (number == 5)

{

if (lenghtM(firstM))

{

takeP = firstM->data;

cout << "Значение взятого предложения равно = ";

printznachP(takeP);

if (lenghtM(firstM) > 1)

firstM = deletefirstM(firstM);

else {

deleteallM(firstM);

lastM = NULL;

firstM = NULL;

}

}

else cout << "Множество пусто.\n";

}

if (number == 6)

{

if (lenghtM(firstM))

{

predl\* read = menuP();

if (sravM(firstM, read))

cout << "Данное предложение уже есть в множестве.\n";

else lastM = addmnoz(firstM, read);

}

else

{

firstM = addfirstM(menuP());

lastM = firstM;

}

}

if (number == 7)

{

if (lenghtM(firstM))

if (sravM(firstM, menuP()))

cout << "Данное предложение присутствует в множестве.\n\n";

else

cout << "Данного предложения в множестве нет.\n\n";

else cout << "Множество пусто.\n";

}

if ((number == 8))

if (lenghtM(firstM)) printmnoz(firstM, lastM); else cout << "Множество пусто\n";

if (number == 9)

{

mode = 0;

if (lenghtM(firstM)) {

deleteallM(firstM);

lastM = NULL;

firstM = NULL;

}

cout << "Работа с множеством завершена.\n";

}

if (number > 9 || number < 1)

cout << "Введите число от 1 до 9" << '\n';

if (number == 101) return firstM;

}

else cout << "Вы не начали работу с множеством. Пункт 1.\n";

if ((number!=8)&&(mode==1)&&(lenghtM(firstM)))

printmnoz(firstM, lastM);

}

} //Меню для работы с множеством

int main()

{

setlocale(LC\_ALL, "Russian");

menuM();

return 0;

}