Материалы презентации предназначены для размещения только для использования студентами кафедры «Компьютерные системы и технологии» НИЯУ МИФИ дневного и вечернего отделений, изучающими курс «Программирование (Алгоритмы и структуры данных)».

Публикация (размещение) данных материалов полностью или частично в электронном или печатном виде в любых других открытых или закрытых изданиях (ресурсах), а также использование их для целей, не связанных с учебным процессом в рамках курса «Программирование (Алгоритмы и структуры данных)» кафедры «КСиТ» НИЯУ МИФИ, без письменного разрешения автора запрещена.

Задача по теме 3 «Обработка таблиц»

Задача

Написать программу для работы с таблицей по запросам оператора.

Вариант б)

– и сама таблица, и информация, относящаяся к элементу таблицы, хранятся во внешней памяти (используется двоичный файл произвольного доступа).
Имя файла вводится по запросу из программы;

– все операции выполняются с таблицей, размещенной в основной памяти;

Задача

Написать программу для работы с таблицей по запросам оператора.

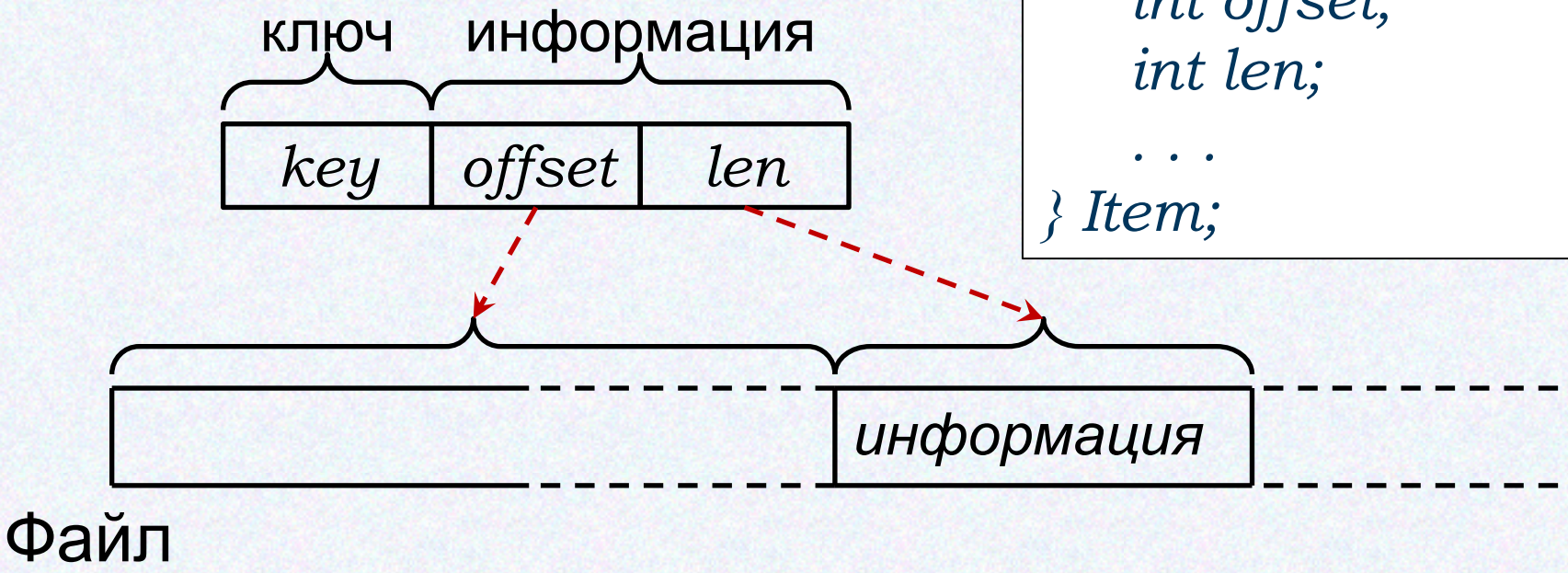
Вариант б)

– таблица считывается из файла (или создается в первый раз) в начале сеанса работы и записывается в файл в конце сеанса работы;

– информация, относящаяся к элементу таблицы, записывается в файл сразу же при выполнении операции включения в таблицу.

Структура элемента таблицы

```
typedef struct Item{
    int key;
    int offset;
    int len;
    . . .
} Item;
```



Длина информации = длина строки + 1!

Структура таблицы

*Item *tab;*

Целевой элемент таблицы

а) таблица – список:

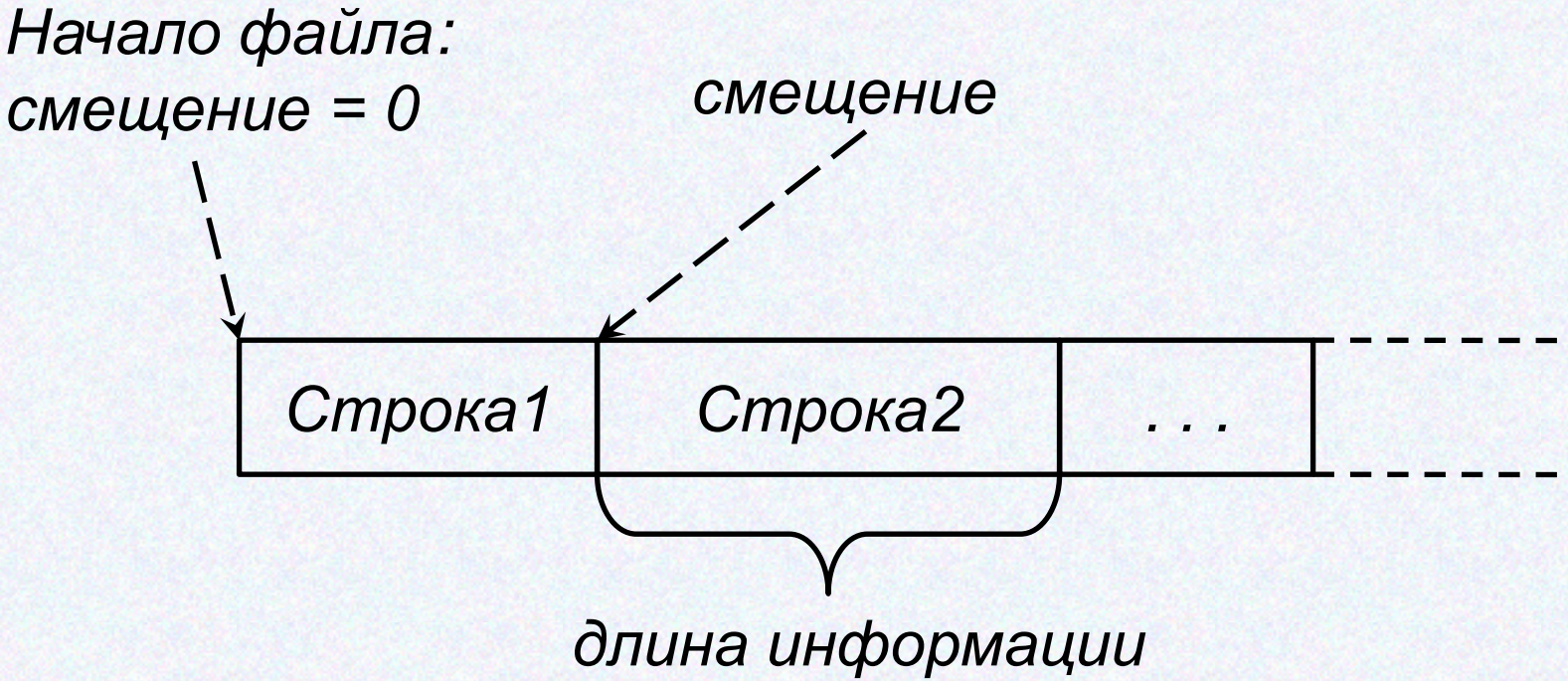
*Item *ptr = . . .;*

б) таблица – вектор:

. . . tab[i] . . .,

*Item *ptr = tab + i;*

Структура файла данных



Запись элемента в таблицу

```
FILE *fd;   // дескриптор файла, открытого на запись
Item *ptr;  // целевой элемент таблицы
char *str;  // информация

ptr->key = ключ элемента;
ptr->size = strlen(str) + 1; // длина информации
```

Запись элемента в таблицу

Позиционирование на конец файла

```
fseek(fd, 0L, SEEK_END);
```

Вычисление смещения в файле

```
ptr->offset = ftell(fd);
```

Запись информации в файл

```
fwrite(str, sizeof(char), ptr->size, fd);
```

Выборка элемента из таблицы

```
FILE *fd;   // дескриптор файла, открытого на чтение
Item *ptr;  // целевой элемент таблицы
char *str;  // информация
```

Выделение памяти под информацию

```
str = (char *)malloc(ptr->size);
```

Позиционирование в файле на требуемую строку данных

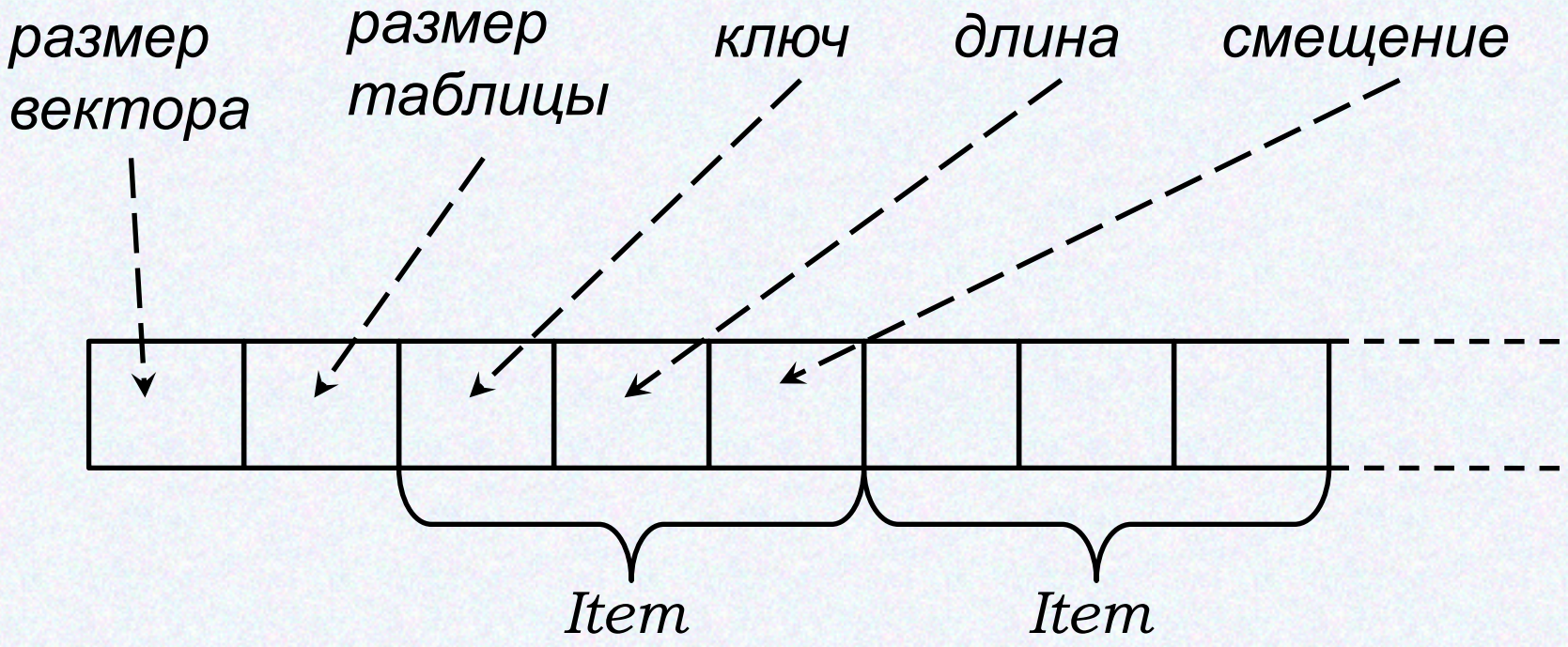
```
fseek(fd, ptr->offset, SEEK_SET);
```

Чтение из файла

```
fread(str, sizeof(char), ptr->size, fd);
```

Таблица – вектор

Структура файла таблицы – вектора



Загрузка таблицы из файла

```
FILE *fd;            // дескриптор файла
char fname[LN];      // имя файла

int size;    // размер вектора
int n;       // размер таблицы
Item *tab;   // сама таблица
```

Загрузка таблицы из файла

Открыть существующий файл на чтение и запись и проверить результат выполнения операции

```
if(fd = fopen(fname, "r+b")){
```

Файл существует, можно загрузить таблицу из файла в оперативную память

Загрузка таблицы из файла

Чтение размера вектора

```
fread(&size, sizeof(int), 1, fd);
```

Выделение памяти под таблицу

```
tab = (Item *)calloc(size, sizeof(Item));
```

Чтение размера таблицы

```
fread(&n, sizeof(int), 1, fd);
```

Чтение таблицы

```
fread(tab, sizeof(Item), n, fd);
}
```

Загрузка таблицы из файла

```
else{
```

Файл не существует, его надо создать и записать в него размер вектора

```
    fd = fopen(fname, "w+b");
    fwrite(&size, sizeof(int), 1, fd);
```

Далее следует выгрузить в файл пустую таблицу, чтобы зарезервировать место

```
}
```

Выгрузка таблицы в файл

```
FILE *fd;          // дескриптор файла
int n;             // размер таблицы
Item *tab;         // сама таблица
```

Пропустить в файле размер вектора

```
fseek(fd, sizeof(int), SEEK_SET);
```

Записать в файл размер таблицы

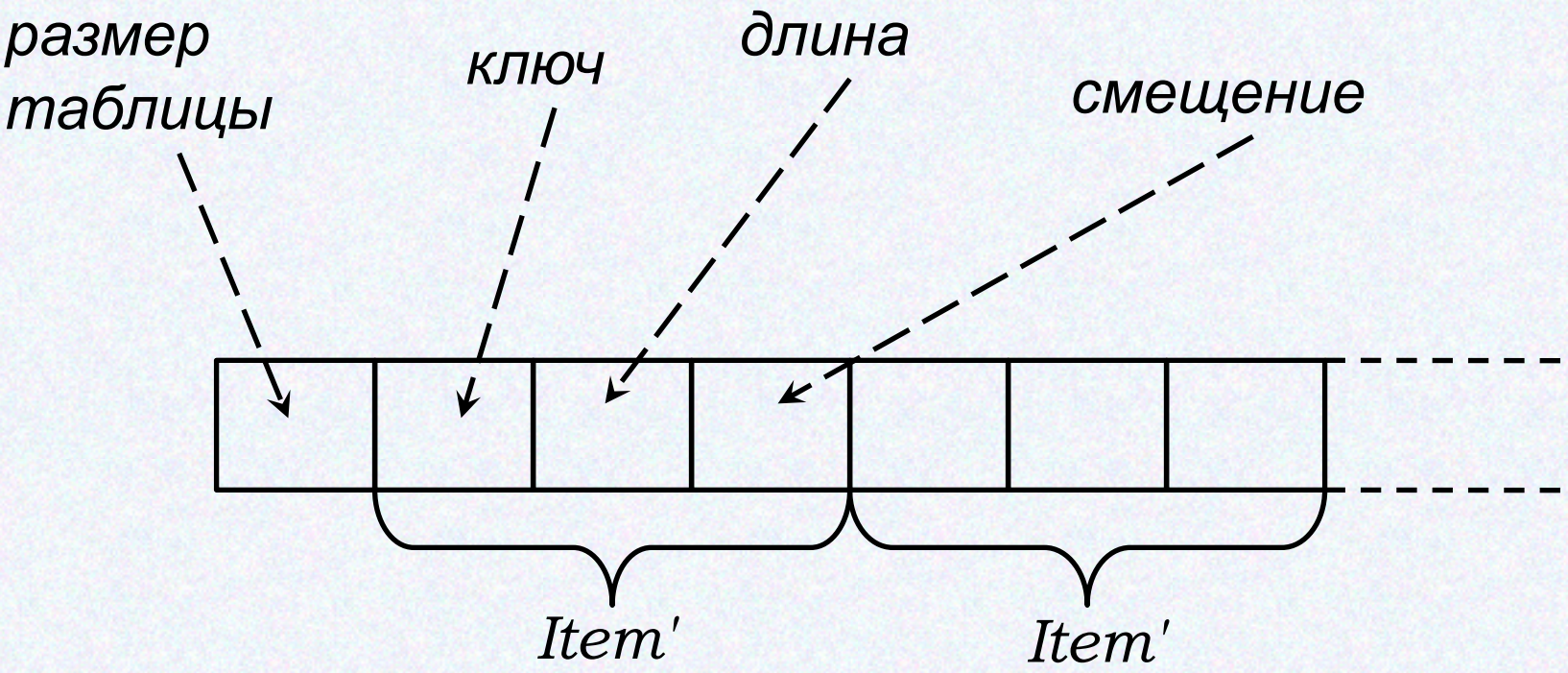
```
fwrite(&n, sizeof(int), 1, fd);
```

Записать в файл саму таблицу

```
fwrite(tab, sizeof(Item), n, fd);
```

Таблица – список

Структура файла таблицы – списка



Загрузка таблицы из файла

```
FILE *fd;              // дескриптор файла
char fname[LN];        // имя файла

int n;          // размер таблицы
Item *tab = NULL;      // сама таблица
Item *last = NULL;     // последний элемент списка
```

Загрузка таблицы из файла

Открыть существующий файл на чтение и проверить результат выполнения операции

```
if(fd = fopen(fname, “rb")){
```

Файл существует, можно загрузить таблицу из файла в оперативную память

Чтение размера таблицы

```
fread(&n, sizeof(int), 1, fd);
```

Загрузка таблицы из файла

Чтение элементов таблицы и занесение их в список

```
while(--n >= 0) {
    Item *cur = (Item *) calloc(1, sizeof(Item);
    fread(&cur->key, sizeof(int), 1, fd);
    fread(&cur->size, sizeof(int), 1, fd);
    fread(&cur->offset, sizeof(int), 1, fd);
```

Загрузка таблицы из файла

Добавление элемента в список

```
if(tab == NULL)
        tab = cur;
else
        last->next = cur;
last = cur;
}
```

Выгрузка таблицы в файл

```
FILE *fd;              // дескриптор файла
char fname[LN];        // имя файла
int n = 0;             // размер таблицы
Item *tab;             // сама таблица
```

Создать новый файл

```
if((fd = fopen(fname, “wb")) == NULL)
```

Вывести сообщение об ошибке и завершить программу

Выгрузка таблицы в файл

Записать в файл саму таблицу

```
for(Item *cur = tab; cur; cur = cur->next){
    ++n;        // текущий размер таблицы
    fread(&cur->key, sizeof(int), 1, fd);
    fread(&cur->size, sizeof(int), 1, fd);
    fread(&cur->offset, sizeof(int), 1, fd);
}
```

Записать в файл размер таблицы

```
fseek(fd, 0, SEEK_SET);
fwrite(&n, sizeof(int), 1, fd);
```