Семинар
Ввод-вывод в языке Си

Ввод-вывод в языке Си

Особенность языка Си – отсутствие заранее спланированных структур файлов. Все файлы рассматриваются как неструктурированная последовательность байтов. При таком подходе удалось распространить понятие файла на все устройства. Одни и те же функции применимы и для устройств, и для файлов.

Ввод-вывод в языке Си

3 уровня ввода-вывода в языке Си:

1. Потоковый ввод-вывод
2. Ввод-вывод нижнего уровня
3. Ввод-вывод для консоли и портов (зависит от операционной системы)

Потоковый ввод-вывод

- Обмен данными производится побайтно
- Такой ввод-вывод возможен для устройств побайтового обмена (принтер, дисплей), так и для файлов на диске (хотя диск – устройство поблочного обмена, за одно обращение происходит считывание или запись фиксированной порции данных)
- При вводе и выводе данных с диска / на диск данные накапливаются в буфере

Потоковый ввод-вывод

Поток – это файл вместе с предоставляемыми средствами буферизации.

Возможные действия при работе с потоком:

- открывать и закрывать потоки (связывать указатели на потоки с конкретными файлами)
- вводить и выводить: символ, строку, форматированные данные, порцию данных
- анализировать ошибки и достижение конца файла
- …

Потоковый ввод-вывод

Возможные действия при работе с потоком (продолжение):

- управлять буферизацией потока и размером буфера
- получать и устанавливать указатель текущей позиции в потоке

Для использования функций библиотеки ввода-вывода в Си необходимо включить в программу файл *stdio.h*

Потоковый ввод-вывод

1. Открытие и закрытие потока

Для работы с потоком в программе необходимо создать экземпляр структуры FILE. При открытии потока в программу возвращается указатель на поток, являющийся указателем на объект структурного типа FILE.

Пример объявления указателя на поток:

```
FILE * myFile;
```

Потоковый ввод-вывод

1. Открытие и закрытие потока

Функция открытия потока:

file = fopen(имя_файла, режим);

Например, для открытия файла для чтения:

file = fopen("t.txt", "r");

Для закрытия потока:

fclose(указатель_на_поток);

fclose(file);

Потоковый ввод-вывод

1. Открытие и закрытие потока

6 режимов открытия файла

w	Новый файл открывается для записи. Если файл существовал, содержимое стирается.
r	Существующий файл открывается только для чтения
a	Файл открывается для добавления информации в конец файла. Если файл не существовал, файл создаётся.
w+	Новый файл открывается как для записи, так и для чтения в любом месте файла. В том числе возможна запись в конец файла. Если файл существовал, содержимое стирается.
r+	Существующий файл открывается как для чтения, так и для записи в любом месте файла, кроме записи в конец файла.
a+	Файл открывается или создаётся для записи или чтения в любом месте, в т.ч. в конце файла. Содержимое существовавшего файла не уничтожается.

Потоковый ввод-вывод

1. Открытие и закрытие потока

В текстовом режиме прочитанная комбинация символов CR(возврат каретки, 13) и LF(перевод строки, 10) преобразуется в один символ новой строки '\n' (10). При записи в поток в текстовом режиме осуществляется обратное преобразование. Для отмены преобразования применяется бинарные режимы, например, "wb", "r+b". В некоторых компиляторах текстовый режим обозначается буквой t.

Потоковый ввод-вывод

1. Открытие и закрытие потока

Основные потоки при выполнении программы:

1. Стандартный поток ввода – stdin
2. Стандартный поток вывода – stdout
3. Стандартный поток вывода сообщений об ошибках – stderr

Соответствия по умолчанию:

stdin – клавиатура

stdout, stderr – дисплей

Потоковый ввод-вывод

2.1 Ввод и вывод символов

Для стандартных потоков:

```
int getchar(void); //читает 1 символ
int putchar(int c); //выводит 1 символ
```

При работе с файлами:

```
int getc(FILE *stream); // или fgetc
int putc(int c, FILE * stream); // или fputc
```

Потоковый ввод-вывод

2.1 Ввод и вывод символов

Функции getchar(), getc(), fgetc() вводят очередной байт информации (символ) в виде значения типа int. Это сделано для успешного распознавания конца файла (EOF). В разных операционных системах константа EOF, определённая в stdio.h, имеет значение 0 или -1.

Потоковый ввод-вывод

2.1 Ввод и вывод символов

При использовании функции getchar() следует помнить, что данная функция получает коды всех символов, введённых с клавиатуры, в т.ч. код символа Enter.

Пример. Программа отображает коды введённых символов.

```
char c;
while(c=getchar()){
    cout<<int(c)<<' '<<endl;
}
```

Потоковый ввод-вывод

2.1 Ввод и вывод символов

Пример. Вывод информации из файла на экран.

```
FILE *file;
char c;
file = fopen("t.txt", "r");
while((c=getc(file))!=EOF)
    putc(c, stdout); // можно putchar(c);
fclose(file);
getchar();
return 0;
```

Потоковый ввод-вывод

2.1 Ввод и вывод символов

Пример. Посимвольное копирование информации из одного файла в другой.

```
FILE *file1, *file2;
char c;
file1 = fopen("source.txt", "r");
if(file1){ // Если source.txt был успешно открыт для чтения
        file2 = fopen("target.txt", "w");
        while((c=fgetc(file1))!=EOF)
                fputc(c, file2);
        fclose(file1);
        fclose(file2);
        puts("Success!");
}
else{puts("Fail");}
```

Потоковый ввод-вывод

2.2 Ввод и вывод строк

Для стандартных потоков:

```
char * gets(char * s); //чтение строки
int puts(char * s); //вывод строки
```

При работе с файлами:

```
char * fgets(char * s, int n, FILE *stream);
int fputs(const char *s, FILE * stream);
```

Потоковый ввод-вывод

2.2 Ввод и вывод строк

Функция fputs() записывает ограниченную символом '\0' строку s в файл, определённый указателем stream. Символ '\0' в файл при этом не записывается. При ошибках возвращается значение EOF.

Функция fgets() читает из определённого указателем stream файла не более (n-1) символов и записывает их в строку s. Дополнительно в конец каждой строки записывается символ конца строки.

Потоковый ввод-вывод

2.2 Ввод и вывод строк

Пример. Программа копирования файлов, использующая аргументы ком. строки.

```
int main(int argc, char* argv[])
{
        char s[256];
        FILE *f1, *f2;
        if (argc != 3)
        {
                printf("\nFormat: copyfile.exe source outcome");
                return 1;
        }
        f1 = fopen(argv[1], "r");
        f2 = fopen(argv[2], "w");
        if(f1 && f2){
                while(fgets(s, 256, f1) != NULL) fputs(s, f2);
                puts("Success!");
        }
        else puts("Fail!");
        fclose(f1);
        fclose(f2);
        return 0;
}
```

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Форматный вывод:
printf(форматная строка, список аргументов);

Возвращаемое значение – число напечатанных символов, в случае ошибки – отрицательное число.

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Форматный вывод:

printf(форматная строка, список аргументов);

Для каждого аргумента должна быть указана только одна спецификация преобразования, имеющая в общем случае вид:

%флаги ширина_поля.точность модификатор спецификатор

Обязателен символ % и спецификатор

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

%флаги ширина_поля.точность модификатор спецификатор

Таблица спецификаторов

d, i	десятичное целое со знаком
u,	десятичное целое без знака
o	восьмеричное целое без знака
x, X (используются проп. буквы)	шестнадцатеричное целое без знака
f	вещественное значение со знаком
e, E	вещественное значение, выводимое в научном виде
g, G	в зависимости от компактности записи, выбирается «f» или «e» («E»)
c	символ
s	строка
p	значение адреса

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

%флаги ширина_поля.точность модификатор спецификатор

Таблица флагов

пробел	перед положительными числами на месте знака используется пробел
+	если выводимое значение имеет знак, то он выводится
#	если этот флаг используется с форматами «o», «x» или «X», то любое ненулевое значение выводится с предшествующими «0», «0x» или «0X»; при использовании с «f», «g», «G» десятичная точка будет выводиться даже если нет дробной части
-	выводимое значение прижимается к левому краю поля

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

%флаги ширина_поля.точность модификатор спецификатор

Таблица модификаторов

h	указывает, что следующий после h спецификатор d, o, x, X применяется к аргументу short или unsigned short
l	указывает, что следующий после l спецификатор d, i, o, x, X применяется к аргументу типа long или unsigned long
L	указывает, что следующий после L спецификатор e, E, f, g, G применяется к аргументу типа long double

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Пример. Ввод десятичного числа, вывод восьмеричного, шестнадцатеричного представления и символа с заданным кодом.

```
int i;
printf("Enter value: ");
scanf("%d", &i);
printf("\nOct = %o, hex = %#X, \
   (char)%+d = %c", i, i, i, (char)i);
```

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

%флаги ширина_поля.точность модификатор спецификатор

Ширина поля задаётся положительным целым числом и определяет минимальное количество позиций, отводимого для представления выводимого значения. Если число символов в выводимом значении меньше, выводимое значение дополняется пробелами. Если ширина поля задана с начальным нулём, не занятые значащими цифрами позиции слева заполняются нулями. Если число символов в выводимом значении больше, печатаются все символы.

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

%флаги ширина_поля.точность модификатор спецификатор

Точность указывается с помощью точки и необязательной последовательности десятичных чисел после неё. Точность задаёт:

- минимальное число цифр, которые могут быть выведены при использовании спецификаторов d, i, o, u, x, X
- число цифр, которые будут выведены после десятичной точки при спецификаторах e, E, f
- максимальное число значащих цифр при спецификаторах g, G
- максимальное число символов, которые будут выведены при спецификаторе s

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Пример использования ширины поля и точности

```
printf("Example 1: %7d %.3f %7.3f %.0f\n", 80, 50.1, 50.1, 50.1);
printf("Example 2: %15s\n", "abcdefg"); // Печатать не менее 15 символов
printf("Example 3: %.5s\n", "abcdefg"); // Печатать не более 5 символов
printf("Example 3: %5.5s\n", "abcdefg"); // Печатать всегда 5 символов
// (лишние символы не выводятся)
```

Вывод:

```
Example 1: 80 50.100 50.100 50
Example 2: abcdefg
Example 3: abcde
Example 3: abcde
```

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Ещё один пример форматного вывода данных

```
int number[3]={1,2,3};//Номер товара
int code[3]={10,25670,120};//Код товара
char name[3][30]=
    {{"lamp"},{"table"},{"very big chair"}};
float price[3]={52.7, 240.0, 824.0};
for(int i=0; i<=2; i++)
    printf("%-3d %5d %-20s %8.3f\n",
    number[i], code[i], name[i], price[i]);
return 0;
```

```
1      10 lamp                 52.700
2   25670 table               240.000
```

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Форматный ввод:
scanf(форматная строка, список аргументов);

Возвращаемое значение – количество введённых полей. Значение EOF возвращается при обнаружении конца файла, значение -1 – при ошибке преобразования данных.

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Форматный ввод:
scanf(форматная строка, список аргументов);

Спецификация преобразования:
% * ширина_поля модификатор спецификатор

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Звёздочка (*), следующая за символом процента, запрещает запись значения, прочитанного из входного потока по адресу, задаваемому аргументом.
Последовательность кодов из входного потока прочитывается функцией, но не преобразуется и не записывается в переменную, определенную очередным аргументом.

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Пример.

```
int i, kol;
char c, str[80];
kol=scanf("code: %d %*s %c %s",&i,&c,str);
printf("\ni=%d c=\'%c\' str=%s", i, c, str);
printf("\n%d values are entered", kol);
```

Вывод:

```
code: 555 mystring1 t MYSTRING2
i=555 c='t' str=MYSTRING2
3 values are entered
```

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Пример. Программа с возможностью многократного ввода числа.

```
int number;
printf("Enter number: ");
while(scanf("%d", &number) != 1){
    //Освобождение входного потока:
    while(getchar()!='\n');
    printf("Error. Enter number:");
}
printf("Your number=%d", number);
```

Потоковый ввод-вывод

2.3 Форматный ввод-вывод

Для форматного ввода-вывода при работе с файлами применяются функции fprintf, fscanf:

```
int fprintf(указатель_на_поток,
форматная_строка, список_переменных);

int fscanf(указатель_на_поток,
форматная_строка,
список_адресов_переменных);
```

Потоковый ввод-вывод

2.4 Позиционирование в потоке

До этого момента мы считывали или записывали данные только последовательно.

При открытии файла в режимах «r» и «w» указатель позиции чтения/записи устанавливается на начало потока, при открытии в режиме «a» – в конец потока.

Потоковый ввод-вывод

2.4 Позиционирование в потоке

При выполнении каждой операции ввода-вывода указатель текущей позиции в потоке перемещается на новую текущую позицию в соответствии с числом прочитанных/записанных байтов.

Потоковый ввод-вывод

2.4 Позиционирование в потоке

Позиционирование в потоке может изменить функция fseek:

int fseek(указатель_на_поток, смещение, начало_отсчёта);

Если функция fseek отработала удачно, возвращаемое значение равно 0.

Потоковый ввод-вывод

2.4 Позиционирование в потоке

int fseek(указатель_на_поток, смещение, начало_отсчёта);

Смещение – число типа long (положительное или отрицательное), определяющее смещение в потоке в байтах.

Начало_отсчёта:

- SEEK_SET (начало файла)
- SEEK_CUR (текущая позиция)
- SEEK_END (конец файла)

Потоковый ввод-вывод

2.4 Позиционирование в потоке

int fseek(указатель_на_поток, смещение, начало_отсчёта);

Смещение – число типа long (положительное или отрицательное), определяющее смещение в потоке в байтах.

Начало_отсчёта:

- SEEK_SET (начало файла)
- SEEK_CUR (текущая позиция)
- SEEK_END (конец файла)

Потоковый ввод-вывод

Задания.

1. В заданном файле подсчитать количество символов английского алфавита, пробелов и цифр.
2. В заданном файле подсчитать количество слов. Считать, что перед словом могут следовать только символ табуляции, пробел, или символ '\n'. Также нужно учесть, что слово может быть расположено в самом начале файла.

Ввод-вывод нижнего уровня

- Использование функций ввода-вывода операционной системы непосредственно
- Не выполняются буферизация и форматирование данных
- Не гарантируется успешность переноса программы из одной ОС в другую

Ввод-вывод нижнего уровня

Основные функции ввода-вывода нижнего уровня:

- open()/close()
- creat()
- read()/write()
- sopen()
- eof()
- lseek()
- tell()

Ввод-вывод нижнего уровня

1. Открытие/закрытие файла

Открытие:
int myFile; // используется дескриптор int!
myFile = open(имя_файла, флаги, права)

Закрытие:
close(дескриптор_файла);
//при успешном закрытии файла возвращ. 0
//в случае ошибки возвращается -1

Ввод-вывод нижнего уровня

Список флагов

O_APPEND	Открыть для записи в конец файла
O_BINARY	Открыть файл в бинарном режиме
O_CREAT	Создать и открыть новый файл
O_EXCL	Если указан с O_CREAT, и файл уже существует, то функция открытия файла завершается с ошибкой
O_RDONLY	Открыть файл только для чтения
O_RDWR	Открыть файл и для чтения, и для записи
O_TEXT	Открыть файл в текстовом режиме
O_TRUNC	Открыть существующий файл и стереть его содержимое

Ввод-вывод нижнего уровня

Права доступа в MS-DOS и Windows:
S_IWRITE – разрешить запись в файл
S_IREAD – разрешить чтение из файла
S_IREAD | S_IWRITE – разрешить и чтение, и запись

Ввод-вывод нижнего уровня

В UNIX права доступа устанавливаются независимо для трёх категорий:

- Владелец файла
- Участник группы пользователей
- Прочие пользователи

Строка прав доступа в UNIX состоит из 9 символов, каждой группе пользователей отводится по 3 символа.

Ввод-вывод нижнего уровня

r – разрешено чтение из файла
w – разрешена запись в файл
x – разрешено выполнение файла

Пример:
rwxr-x--x
Для владельца файла разрешены все действия, для участника группы пользователей – только чтение и выполнение, для прочих – выполнение.

Ввод-вывод нижнего уровня

Примеры:

```
//Открытие для чтения:
file = open("1.txt", O_RDONLY);
//Для записи новых данных с правами rw----:
file = open("2.txt",
O_WRONLY|O_CREAT|O_TRUNC, 0600);
//Для добавления данных с правами rw----:
file = open("3.txt",
O_WRONLY|O_APPEND|O_CREAT, 0600);
```

Ввод-вывод нижнего уровня

2. Чтение и запись данных

```
int read(int file, char * buf, unsigned int count);
int write(int file, char * buf, unsigned int count);
```

При возникновении ошибок функции возвращают -1