Операции с файлами и потоками

1. Открытие файла на чтение или запись – создание потока
2. Закрытие потока
3. Последовательное чтение из потока
4. Последовательная запись в поток
5. Смещение в потоке (не всегда возможно)

Операции со стандартными потоками ввода/вывода

int printf (const char *, …);

Возвращает количество напечатанных символов

int scanf (const char *, …);

Возвращает количество считанных аргументов

Работа с файлами

```
#include <stdio.h>

int main (void)
{
        //открываем файл на запись – создаем поток
        FILE * file = fopen (“hello.txt”, “wt”);
        //пишем в поток
        fprintf (file, “Hello, world!\n”);
        //закрываем поток
        fclose (file);
        return 0;
}
```

Открытие файла

FILE * fopen (char const * path, char const * mode);

path – имя файла

mode – режим открытия

Режимы открытия в fopen

r – открытие на чтение. В случае отсутствия файла возвращается нулевой указатель.

w – открытие на запись. Если файл не существует, он создается. Если существует, по уничтожается и создается заново.

a – открытие на запись. Если файл не существует, он создается. Если существует, но дополняется.

r+, w+, a+ - открытие на чтение/запись

Режимы открытия в fopen

t – открытие в текстовом режиме

b – открытие в двоичном режиме

Проверка и закрытие потока

int fclose (FILE * file);
Закрытие потока
Возвращает 0 в случае успеха и не 0 в случае ошибки

int feof (FILE * file);
Проверка потока
Возвращает не 0, если достигнут конец файла

Ввод/вывод в текстовом режиме

int fprintf (FILE * file, const char *, …);
Возвращает количество напечатанных символов

int fscanf (FILE * file, const char *, …);
Возвращает количество считанных аргументов

char* fgets (char * str, int size, FILE * file);
Читает строку размера не более чем size – 1
Возвращает str в случае успеха и 0 в противном случае

Построчное чтение файла

```
#include <stdio.h>

int main (int argc, char ** argv)
{
        if (argc < 2) return 1;
        FILE * file = fopen (argv[1], "rt");
         char str [1000];
        int i = 0;
        while (!feof (file))
        {
                //fscanf (file, "%s", str);
                fgets (str, 1000, file);
                printf ("str%.2d: %s", i++, str);
        }
        fclose (file);
        return 0;
}
```

Ввод/вывод в бинарном режиме

size_t fread(void *ptr, size_t size, size_t n, FILE *file);
читает n элементов размером size байт (size*n байт)
из потока и записывает их по указателю

size_t fwrite(const void *ptr, size_t size, size_t n, FILE *file);
пишет в поток n элементов размером size (size*n байт)
байт в поток из указателя

Предопределенные потоки

STDOUT – стандартный поток вывода
printf (…) эквивалентно fprintf (STDOUT, …)

STDIN – стандартный поток ввода
scanf (…) эквивалентно fscanf (STDIN, …)

STDERR – стандартный поток ошибок

Смещение в потоке

int fseek(FILE *file, long offset, int whence);

offset – смещение в байтах

whence – указывает, откуда отсчитывается смещение

SEEK_SET – относительно начала файла

SEEK_CUR – относительно текущей позиции

SEEK_END – относительно конца файла

возвращает 0 в случае успешного исполнения

long ftell(FILE *file);

возвращает смещение в потоке относительно начала

Смещение в потоке

```
#include <stdio.h>
int main (int argc, char ** argv)
{
        if (argc < 2)
        {
                fprintf (STDERR, “Not enough arguments\n”);
                return 1;
        }
        FILE * file = fopen (argv[1], "rb");
        fseek (file, 0, SEEK_END);
        printf ("Length of \"%s\" == %ld\n", argv[1], ftell (file));
        fclose (file);
        return 0;
}
```

Ссылки (references)

Ссылка – это альтернативное имя объекта.
При инициализации ссылки ей сопоставляется объект, после этого с ней можно работать как с обычной переменной

```
int a = 0;
int &aref = a;          //инициализация ссылки
aref = 1;               //a = 1
```

```
type & имя_ссылки = lvalue;
```

Передача параметров по ссылке

```
void initDouble (double *dptr)
{
        *dptr = 0;
}

int main (void)
{
        double a;
        initDouble (&a);
        //a == 0
}
```

```
void initDouble (double &dref)
{
        dref = 0;
}

int main (void)
{
        double a;
        initDouble (a);
        //a == 0
}
```

Константные объекты

type const – объекты данного типа являются немодифицируемыми (константными), возможна только их инициализация.

```
int const a = 10;      //инициализация
/*a = 20;*/            //ошибка – модификация недопустима
/*int &aref = a;*/     //ошибка – попытка создать
                       //модифицируемую ссылку
int const &aref = a;

/*int *aptr = &a;*/   //ошибка – попытка создать
                       //модифицируемый указатель
int const *aptr = &a;
```

Константные объекты

Различайте объекты и их имена. Объект – это форматированная область памяти,
у нее может быть много имен.

1. Объект может иметь модифицируемые и константные имена.
2. Модифицируемое имя всегда может быть использовано, как константное, но не наоборот!
3. В сложных типах (int *) const всегда применяется к левой части типа:

```
int const * ptrToConst; //указатель на константу
const int * ptrToConst; //указатель на константу
int * const ptrToConst; //константный указатель
```

const в параметрах функций

```
struct Circle {/*поля*/};

Circle initCircle (void);
void initCircle (Circle * arg);

void drawCircle (Circle arg);
void drawCircle (Circle *arg);
```

```
struct Circle {/*поля*/};

void initCircle (Circle & arg);

void drawCircle (Circle const & arg);
```

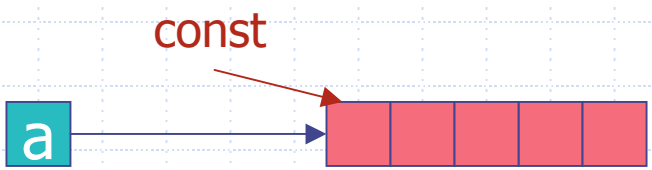
Сложные типы

```
int ** a;                  //указатель на указатель

int * a [];                //массив указателей

int a = new int [10];
int char *& aref=a;  //ссылка на указатель

int char &* a;             //указатель на ссылку – не бывает!

int & a [];                //массив ссылок – не бывает!
```

Сложные типы и const

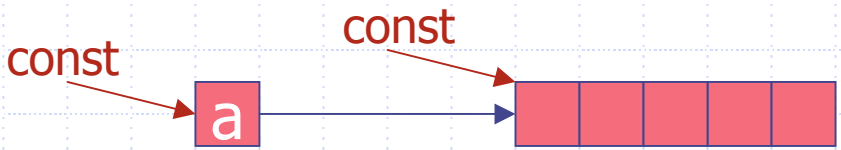
```
char* const a;          //константный указатель
```


```
char const * a;         //указатель на константу
const char * a;         //то же самое. Допускается!
```

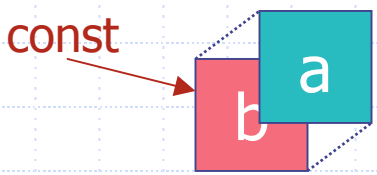


Сложные типы и const

```
char const * const a;
```




```
char a;
char const & b;        //ссылка на константу
```



Размер массивов и const

```
int const size = 10;
int arr [size];          //допустимо
```

```
int const size1 = 10;
int const size = size1;
int arr [size];          //допустимо
```

```
int size1 = 10;
int const size = size1;
int arr [size];          //недопустимо в старом стандарте
```