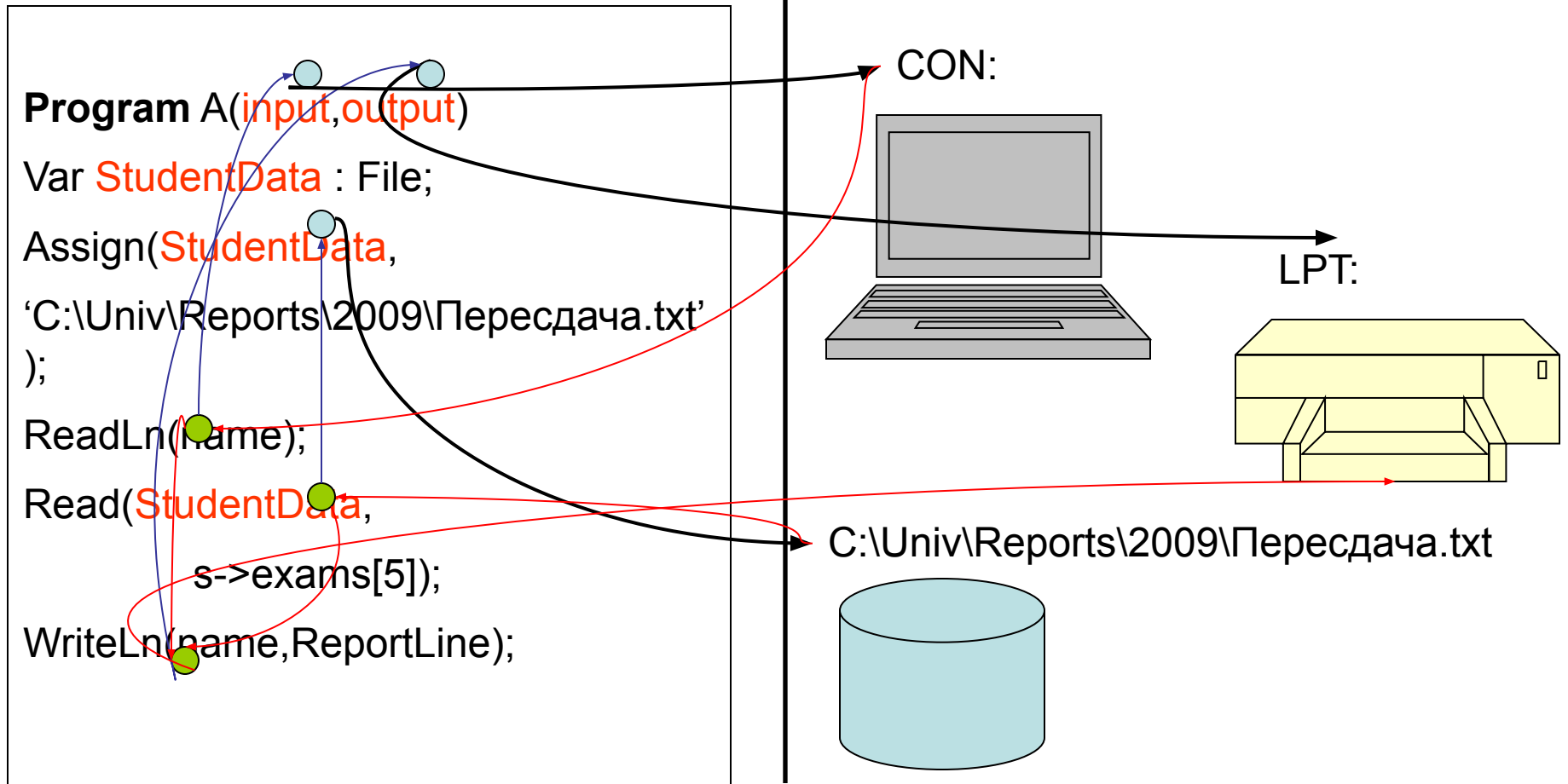


Ввод-вывод - файлы

Программа: логические файлы

Операционная и файловая системы: физические файлы



Логические файлы

- Являются объектами программы и обеспечивают связь с физическими файлами
- Скрывают особенности реализации разных видов файлов
- Могут в разные моменты времени исполнения быть связаны с разными физическими файлами

Ввод/вывод – СВЯЗЬ С ЯЗЫКОМ

- Специальные конструкции языка (Fortran)

```
      READ (f,2) (X(I), I=1,100)  
2 ←  FORMAT (16F5,1)
```

- Псевдо-процедуры (Pascal)

```
Write(f, x:6:2, ' + ', y:6:2,  
      ' = ', x+y :7:2);  
WriteLn(f);
```

- Элементарные процедуры (Modula-2)

```
FWriteFloat(f, x, 6, 2);  
FWriteString(f, ' + ');  
FWriteFloat(f, y, 6, 2);  
FWriteString(f, ' = ');  
FWriteFloat(f, y, 7, 2);  
FWriteLn(f);
```

- Процедуры форматного вывода (C)

```
fprintf(f,  
      "%6.2f + %6.2f = %7.2f\n",  
      x, y, x+y);
```

Ввод/вывод – СВЯЗЬ С ЯЗЫКОМ

- Специальные конструкции языка
 - Богатые (но фиксированные) возможности
 - Нет накладных расходов на вызов
- Библиотечные процедуры
 - Минимизация языковых концепций
 - Подключается только при необходимости

Низкоуровневый ввод/вывод

```
#include <fcntl.h>
```

- // создание файла
int creat(char *filename, int permission);
- // открытие файла
int open(char *filename, int access, int permission);
- // чтение из файла в буфер
int read(int handle, void *buffer, int nbyte);
- // запись из буфера в файл
int write(int handle, void *buffer, int nbyte);
- // установка текущей позиции
long lseek(int handle, long offset, int whence);
- // закрытие файла – освобождение ресурсов
int close(int handle);
- // удаление файла
int unlink(char *filename);

Низкоуровневый ввод/вывод

```
#include <fcntl.h>
int fd;
char buffer[10];
fd = open(
    "C:\Univ\Reports\2009\Пересдача.txt ",
    O_RDONLY | O_TEXT );
lseek(fd,4,SEEK_SET);
read(fd, buffer, 10);
close(fd);
```

fd:

buffer:

о	в	,	5	,	4	,	3	;	С
---	---	---	---	---	---	---	---	---	---

П	е	т	р	о	в	,	5	,	4	,	3	;	С	и	д	о	...
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	-----



Низкоуровневый ввод/вывод

- Системные вызовы – дорогая операция
- Код ответа (м.б. непривычно):
 - 0 – всё нормально
 - номер причины неудачи, иначе
- Поддерживает только вывод байтов и текста; нет чисел, строк и т.д.
- Использовать только в крайнем случае!

Буферизованный ввод/вывод

```
#include <stdio.h>
```

- // открытие файла

```
FILE *fopen(char *filename, char *mode);
```

- mode == “r” – чтение
- mode == “w” – запись
- mode == “a” – дозапись

- // чтение из файла count элементов размера size

```
long fread(void* ptr, long size, long count, FILE * stream);
```

- // запись в файл count элементов размера size

```
long fwrite(void* ptr, long size, long count, FILE * stream);
```

- // установка текущей позиции

```
int fseek(FILE * stream, long offset, int origin);
```

- // установка текущей позиции

```
long ftell(FILE * stream);
```

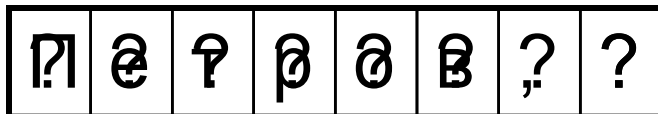
- // закрытие файла – освобождение ресурсов

```
int fclose(FILE * stream);
```

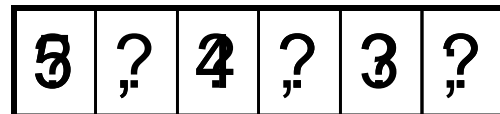

Буферизованный ввод/вывод

```
FILE * f;  
char bname[8], bmarks[6];  
f = fopen("C:\Univ\Reports\2009\Пересдача.txt ", "r");  
fread(bname,7,1,f);  
fread(bmarks,6,1,f);  
fclose(f);
```

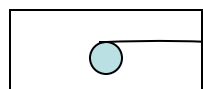
bname:



bmarks:



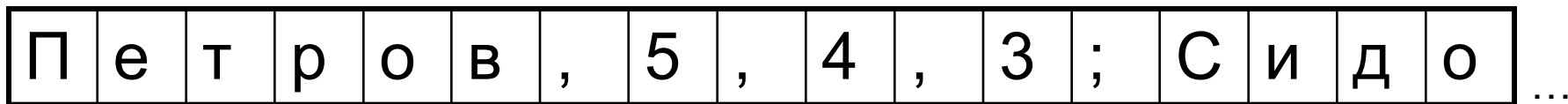
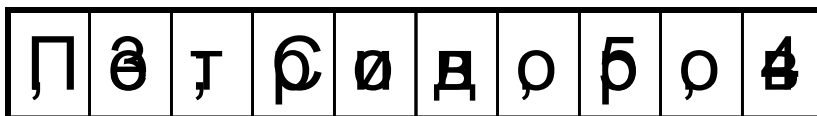
f:



fd:



buffer:



Буферизованный ввод/вывод

- `fread`, `fwrite` могут быть значительно эффективнее `read` и `write`
- `fread`, `fwrite` уместны для работы с «бинарными» данными, для которых точно известен размер
- Типичные ошибки:
 - незакрытие файла
 - чтение из закрытого файла
 - повторное закрытие файла
 - несоответствие размера данных

Посимвольный и построчный ввод/вывод

```
FILE * in, * out;  
in = fopen("in.txt", "r");  
out = fopen("out.txt", "w");  
int c;  
while ((c=fgetc(in) != EOF)  
    fputc(c,out);  
fclose(in);  
fclose(out);
```

```
FILE * in, * out;  
in = fopen("in.txt", "r");  
out = fopen("out.txt", "w");  
char buf[N];  
while ((fgets(buf,N,in)) != NULL)  
    fputs(buf,out);  
fclose(in);  
fclose(out);
```

- fgetc вызывается большее количество раз
- fgets делает анализ строки

Стандартные файлы

- `stdin` – стандартный ввод
- `stdout` – стандартный вывод
- `stderr` – файл ошибок
- Перенаправление ввода/вывода
`My1stProg.exe < StudentData.txt > Report.txt`
- Типичная ошибка
`fclose(stdout);`
(кто создаёт, тот и удаляет)

Форматный ввод/вывод (пример)

Входные данные:

Стипендия 2700
Долг -300
Телефон -500
Одежда -750
Еда -123.50
Долг 500

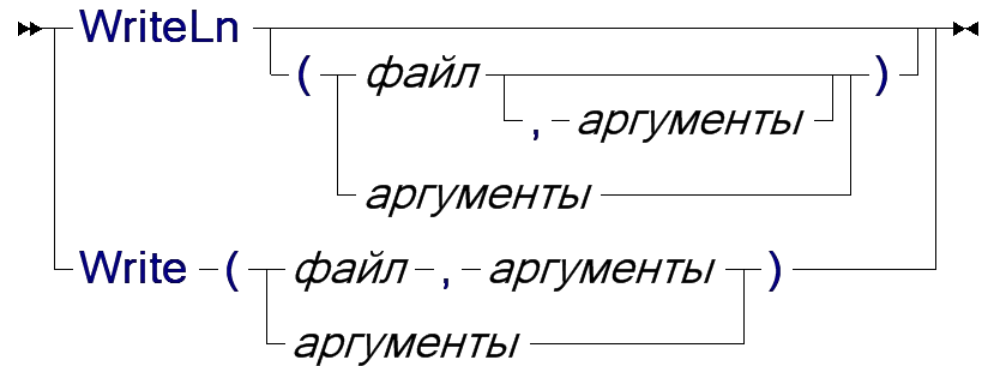
Выходной отчёт:

Приход	Сумма	Баланс	Сумма	Расход
Стипендия	2700.00	2700.00		
		2400.00	300.00	Долг
		1900.00	500.00	Телефон
		1150.00	750.00	Одежда
		826.50	123.50	Еда
Долг	500.00	1326.50		

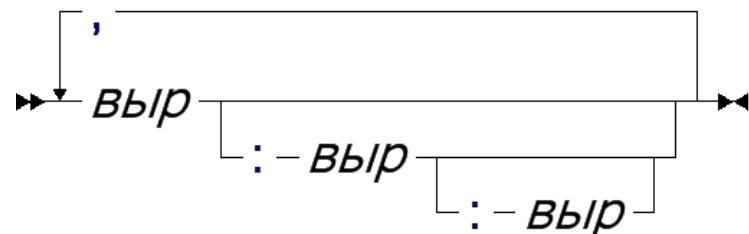
Форматный ввод/вывод (Pascal)

Специальный синтаксис
фактических
параметров в
псевдопроцедурах
`Write`, `WriteLn`

```
WriteLn(out,  
comment:20,  
amount:10:2,  
balance:10:2);
```



аргументы:



ФОРМАТНЫЙ ВВОД/ВЫВОД (C)

- // ВЫВОД В ФАЙЛ

```
int fprintf(FILE * stream, char * format, ... );
```

- // ВЫВОД В stdout

```
int printf(char * format, ... );
```

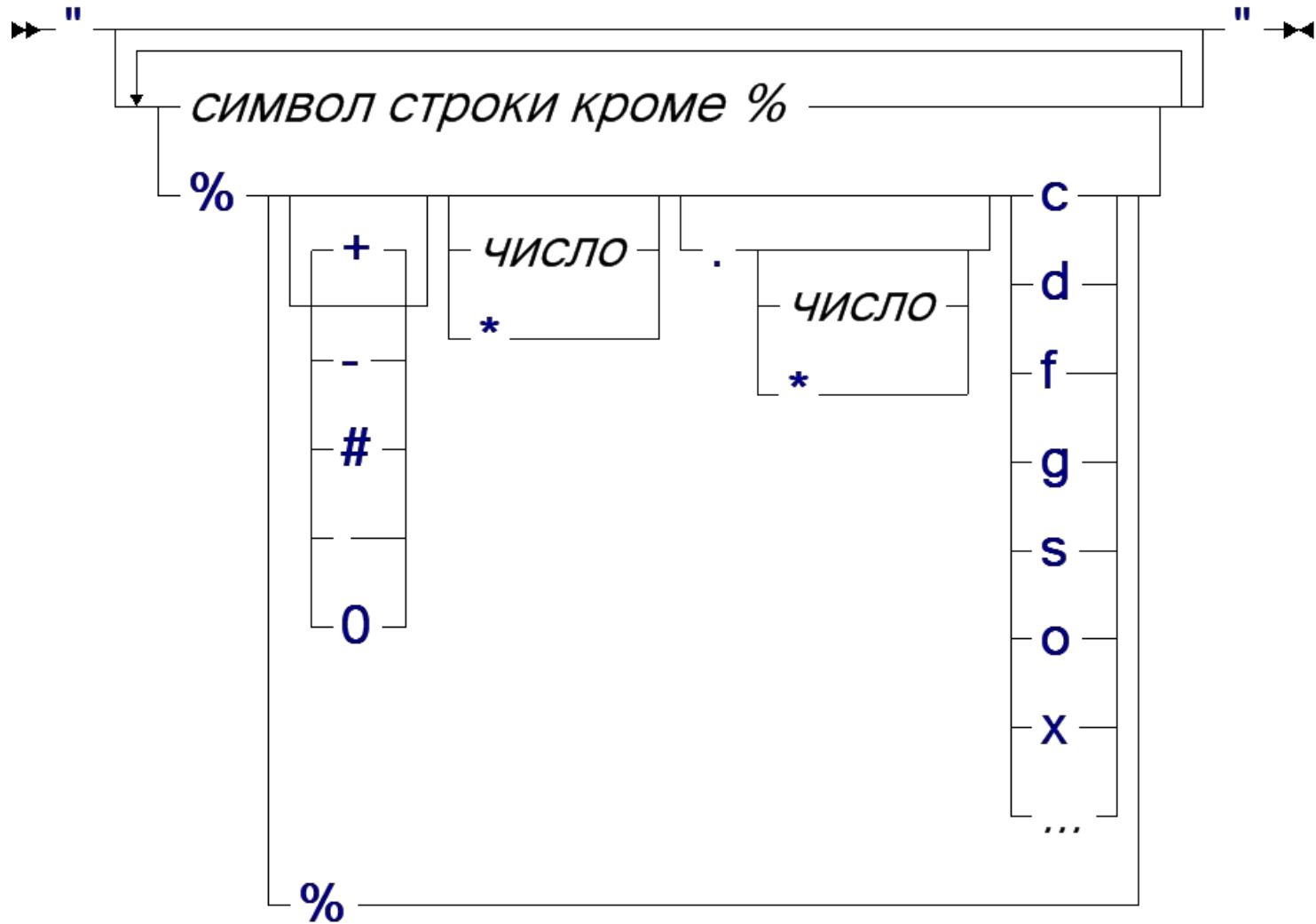
- // ЧТЕНИЕ ИЗ ФАЙЛА

```
int fscanf(FILE * stream, char* format, ... );
```

- // ЧТЕНИЕ ИЗ stdin

```
int scanf(char * format, ... );
```

Формат вывода



ФОРМАТНЫЙ ВВОД/ВЫВОД (пример)

```
void main()
{
    FILE * in = ...
    FILE * out = ...;

    char comment[128];
    float amount;
    float balance = 0.0;

    char header_fmt[] =
        "%20s%10s%10s%10s%20s\n";
    char income_fmt[] =
        "%-20.20s%+10.2f%+10.2f%10s%20s\n";
    char expense_fmt[] =
        "%20s%10s%+10.2f%+10.2f%-20.20s\n";
```

```
    fprintf(out, header_fmt, "Приход", "Сумма",
        "Баланс", "Сумма", "Расход");
    while (fscanf(in, "%s%f",
        comment, &amount) == 2)
    {
        balance += amount;
        if (amount >= 0)
            fprintf(out, income_fmt,
                comment, amount, balance, "", "");
        else
            fprintf(out, expense_fmt,
                "", "", balance, -amount, comment);
    }
    fclose(in);
    fclose(out);
}
```

Форматирование строк

- // «ВЫВОД» В СТРОКУ

```
int sprintf(char * src, char * format, ... );
```

- // «ЧТЕНИЕ» ИЗ СТРОКИ

```
int sscanf(char * src, char* format, ... );
```

sprintf

- Пример: центрированная печать

```
void WriteCenterString(char * s, int p)
{
    char fmt[20];
    int l = strlen(s);
    // если не помещается,
    // увеличиваем p
    if (p < l) p = l;

    l += (p - l) / 2;
    sprintf(fmt, "%%%ds%%%ds", l, p-l);

    // например, при l=4, p=20
    // имеем p2=8, fmt = "%12s%8s"
    printf(fmt, s, "");
}
```

```
void WriteCenterFloat(float x, int p)
{
    // xstr – текстовое представление x
    char xstr[20];
    sprintf(xstr, "%f", x);

    WriteCenterString(xstr,p);
}
```

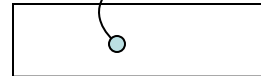
Вывод/вывод указателей

```
struct Person
{
    struct Person * Parent;
    char Name[32];
    unsigned int ChildrenCount;
    struct Person * Children;
} * root, * child;
```

```
fwrite (&root,1,sizeof(root),f);
```

В файл попадёт
4 байта

root:



Parent	
Name	Сергей
ChildrenCount	2
Children	



Parent	
Name	Андрон
ChildrenCount	0
Children	

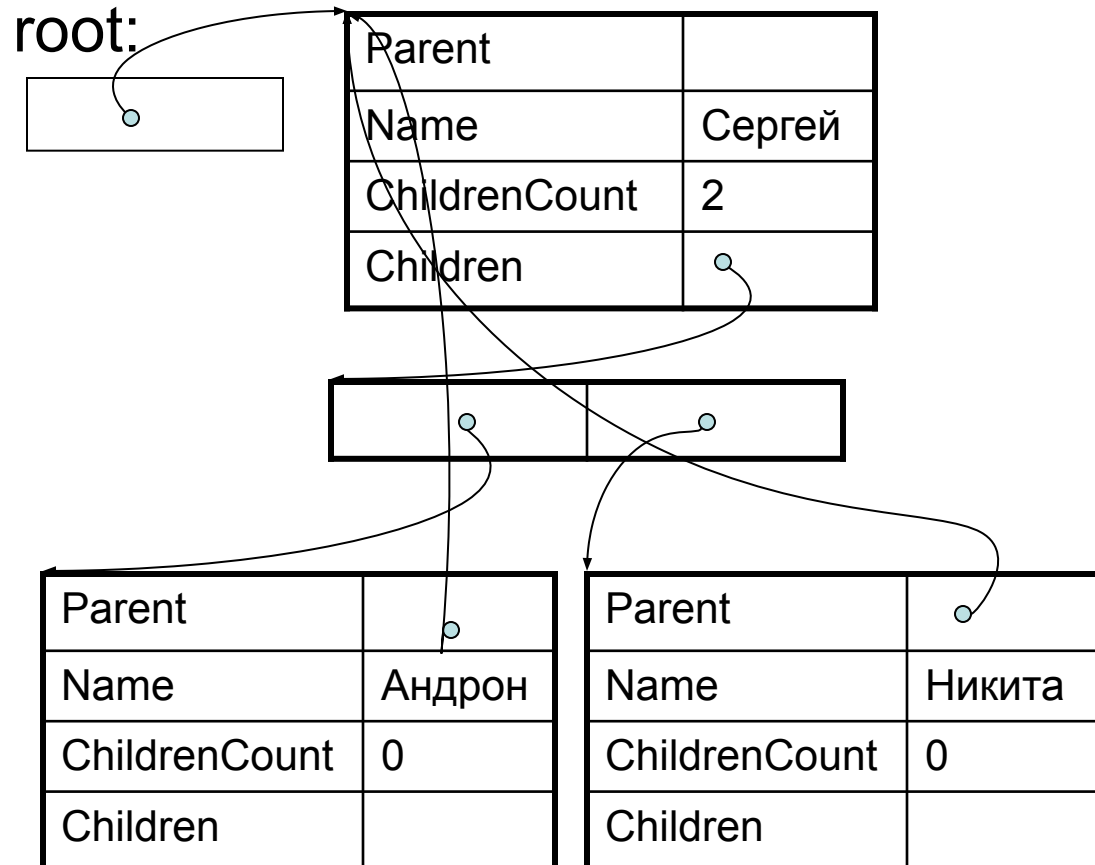
Parent	
Name	Никита
ChildrenCount	0
Children	

Вывод/вывод указателей

```
struct Person
{
    struct Person * Parent;
    char Name[32];
    unsigned int ChildrenCount;
    struct Person * Children;
} * root, * child;
```

```
fwrite(&root, 1, sizeof(root), f);
fwrite(root, 1, sizeof(*root), f);
fwrite(root->Children, 2,
        sizeof(struct Person *), f);
fwrite(root->Children[0], 1,
        sizeof(struct Person), f);
fwrite(root->Children[1], 1,
        sizeof(struct Person), f);
```

Запишется всё, но...



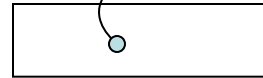
Вывод/вывод указателей

```
struct Person
{
    struct Person * Parent;
    char Name[32];
    unsigned int ChildrenCount;
    struct Person * Children;
} * root, * child;
```

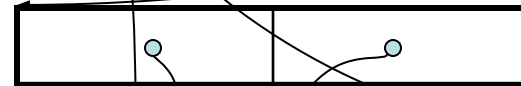
```
freed(&root,1,sizeof(root),f);
freed(root,1,sizeof(*root),f);
freed(root->Children,2,
    sizeof(struct Person *),f);
freed(root->Children[0],1,
    sizeof(struct Person),f);
freed(root->Children[1],1,
    sizeof(struct Person),f);
```

Считаются ссылки на
несуществующие
объекты

root:



Parent	
Name	Сергей
ChildrenCount	2
Children	



Parent	
Name	Андрон
ChildrenCount	0
Children	

Parent	
Name	Никита
ChildrenCount	0
Children	

Ввод/вывод указателей - специализированные процедуры

```
struct Person
{
    struct Person * Parent;
    char Name[32];
    unsigned int ChildrenCount;
    struct Person * Children;
} * root, * child;
```

- Сохраняется лишнее
- Предполагается отсутствие циклов

```
void SavePerson(struct Person *p, FILE * f)
{
    fwrite(p, 1, sizeof(*p), f);
    for (int i = 0; i < p->ChildrenCount; i++)
        SavePerson(p->Children[i], f);
}

struct Person * LoadPerson(FILE * f)
{
    struct Person * p;
    new(p);
    fread(p, 1, sizeof(*p), f);
    if (p->ChildrenCount > 0)
    {
        newarray(p->Children, p->ChildrenCount);
        for (int i = 0; i < p->ChildrenCount; i++)
        {
            p->Children[i] = LoadPerson(f);
            p->children[i]->Parent = p;
        }
    }
    return p;
}
```

Ввод/вывод указателей - универсальные процедуры

- Требуется знание о типе данных во время исполнения
- Сериализация
 - замыкание – сохранение всего, что доступно
 - соответствие между реальными и сохранёнными указателями
- Требуется языковая поддержка
 - пример PS Algol: нет разницы между внутренними и внешними данными.