Структури от данни и програмиране

Лекция 7

Файлове

Мотивация

- Досега разглежданите програми съхраняваха своите данни единствено в оперативната памет
  - След приключване на изпълнението на програмата данните се губят
- Решение: записване на данните във файлове
  - Обикновено върху траен носител, например харддиск

Файлове

- В повечето съвременни операционни системи (ОС) файловете са организирани като едномерен масив от байтове
  - Какво е байт?

Формат на файловете

- Определя как информацията да бъде представена в цифров вид
  - (така че да може после да бъде възстановена при четене на файла)
- ОС не налага никакви ограничения

Примерни файлови формати

- Plain text
  - Обикновено всеки символ се представя с един байт – ASCII кода на този символ
    - Освен видимите има и специални, напр. символът за нов ред
  - ASCII таблица – вж. следващия слайд
  - Масова употреба
    - txt файлове
    - source код на приложения (напр. cpp и h файлове)
    - HTML
    - XML
    - конфигурационни файлове и много други
  - 256 символа не са достатъчни, затова в днешно време се използва и Unicode (UTF-16, UTF-8...)
  - Важно предимство: могат лесно да се четат и манипулират с обикновен текстов редактор като Notepad

ASCII таблица

- Кодове 0-127 (7 бита)
- За 128-255 има различни варианти Extended ASCII
  - Например за кирилица – Windows 1251
- Символите от 0 до 31 са по-специални
- Малките и главните английски букви имат различни кодове
- Кодовете на цифрите от 0 до 9 не са от 0 до 9
- Кодовете на съседни букви (цифри) също са съседни

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Текстов файл - пример

- Нека имаме текстов файл със следното съдържание:

```
It's easier
than you think.
```

- Файлът може да се представи със следната последователност от байтове (в шестнайсетична бройна система):

```
49 74 27 73 20 65 61 73 69 65 72 0a 74 68
61 6e 20 79 6f 75 20 74 68 69 6e 6b 2e
```

- Особеност: символът за нов ред е различен в различните ОС, например:
  - 13 10 (0d 0a, или \r\n, или CR LF) при Windows и DOS
  - 10 (0a) при Linux, Unix и MacOS X
  - 13 (0d) при старите MacOS

Примерни файлови формати (2)

- Bitmap
  - Прост формат за съхранение на растерни изображения
    - Изображението представлява матрица от пиксели, всеки от които има определен цвят
  - Първите (много често) 54 байта (header) съдържат информация като широчина и височина в пиксели, брой битове за всеки пиксел и т.н.
  - Следва информация за цвета на всеки пиксел, представена с указания брой битове
    - Ако изображението е 24-битово, всеки пиксел се представя с 3 байта – по един за червения, зеления и синия цвят

Допълнителен материал

- Демонстрация: разглеждане на файлове от различен формат с текстов и с шестнайсетичен редактор
- Разпознаване на файлов формат по съдържанието му
  - Ако не е текстов файл, обикновено първите няколко байта нарочно идентифицират формата
- Файлови разширения в Windows, задаване на програма, с която да се отвори даден тип файл
  - Какво става при смяна на разширението на файл?

Допълнителен материал (2)

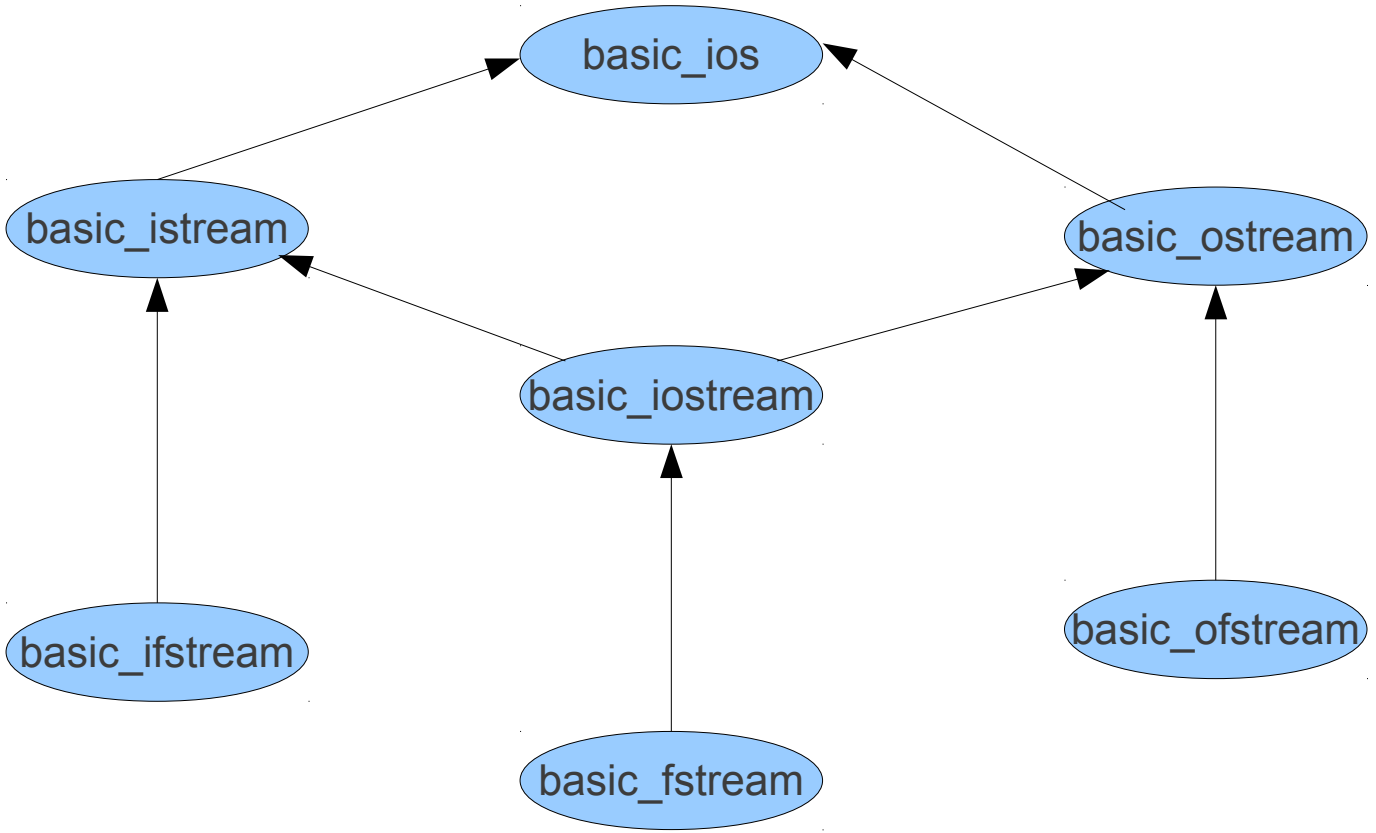
- Файлова система
  - Пространството за съхранение на данни (на твърд диск, флаш памет, оптичен диск и т.н.) също е масив от байтове
  - Цели: да може да се съхраняват много файлове, да има директорийна структура и др.
  - Допълнителната информация заема част от пространството за съхранение, затова празен носител с капацитет X байта има по-малко от X байта използваемо пространство
  - Примерни ФС: FAT, NTFS, Ext3
    - Изучават се в курса по операционни системи

Класификация на файловете

Текстови файлове	Двоични файлове
Интерпретиране на данните във файла като текст (ASCII, Unicode или др.) (прилича на символен низ)	Неформатиран (суров) вход и изход
Последователен достъп – за да достъпим n-ти елемент, трябва да прочетем предходните n-1 елемента	Позволяват пряк достъп
Еднократно обхождане	Многократно обхождане
Файлов формат: plain text	Примерни файлови формати: изпълними файлове (exe), doc (Word), bitmap, JPEG, MP3, zip и др.

Работа с файлове в C++

Поточна йерархия

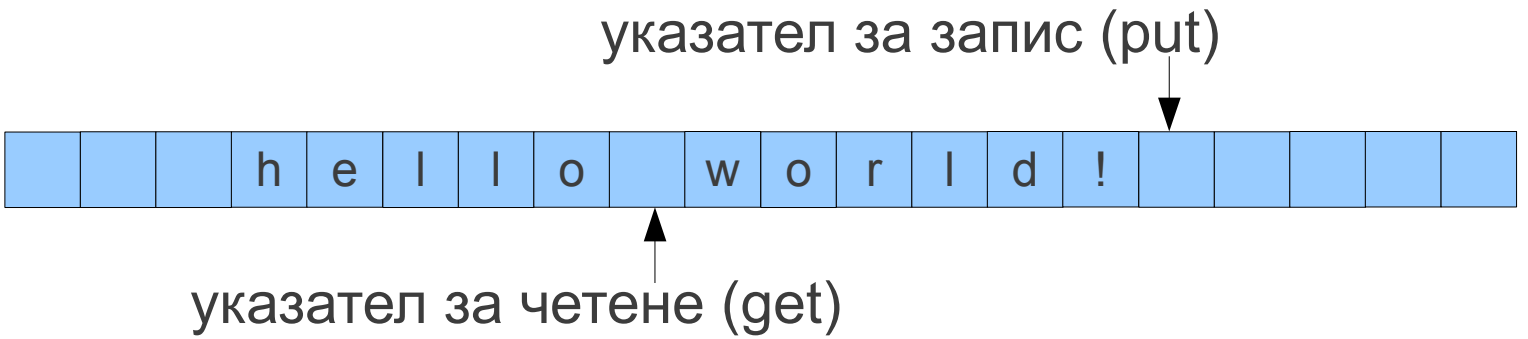


Класове за работа с файлове, използвайки потоци

- `#include <fstream>`
- `ifstream` – поток за четене
  - Връзка с предходната диаграма: класът е дефиниран като
    `typedef basic_ifstream<char>`
- `ofstream` – поток за писане
- `fstream` – поток за четене и писане

Указатели към текущата позиция за четене и за писане

- Методи:
  - tellg – връща текущата позиция на get
  - seekg – премества get на указаната позиция
  - tellp, seekp – аналогично за put



Отваряне на файл

- `<клас> <име>(<път_до_файла>`
  `[, <режим_на_отваряне>]);`
- или

  `<клас> <име>;`
  `<име>.open(<път_до_файла>`
  `[, <режим_на_отваряне>]);`
- Пример:

  `ifstream f("test.txt");`

Режими на отваряне на файл

- Статични константи `ios::<константа>`

константа	предназначение
in	Отваря файл за извличане (подразбира се за класа ifstream).
out	Отваря файл за вмъкване и установява указателя put в нача-лото на файла. Операцията вмъкване може да се извършва на произволно място във файла. Подразбира се за класа ofstream.
app	Отваря файл за вмъкване в края му.
ate	Отваря файл за вмъкване и установява указателя put в края на файла.
trunc	Изтрива съдържанието на файла, ако той съществува. Това действие е подразбиращо се за ios::out.
binary	Отваря файл за двоичен (не текстов) вход или изход.

- Може да се комбинират с побитово ИЛИ (“|”)

Подразбиращи се режими

- Ако не се укаже режим на отваряне, се използват следните:

ifstream	ios::in
ofstream	ios::out
fstream	ios::in | ios::out

Допълнителен материал: път до файл (file path)

- Ако се посочи само име на файл, той ще бъде търсен в текущата директория
- Може да се посочи абсолютен път:

```
"C:\\My Documents\\file.txt"
```

  - Под Windows разделителят е '\', а под Linux - '/'
  - Тъй като '\' се използва за escape-ване на някои символи, напр. '\n' за нов ред, самата '\' също трябва да се escape-не - '\\'
- Може да се посочи относителен път спрямо текущата директория:

```
"../dir1/dir2/file.txt"
```

Четене от текстов файл

- Конструиране на ifstream обект
- Използват се същите начини за четене като при cin – operator>>, getline и т.н.
  - cin е обект от клас istream – базов клас на ifstream
- Затваряне на файла с метода close()
  - Ако не го извикаме, автоматично се извиква от деструктора на ifstream
  - Не трябва да се оставя отворен файл, когато не е необходимо – ще попречи на работата на други програми, напр. няма да може да бъде изтрит
- Пример

Писане в текстов файл

- Конструиране на ofstream обект
  - ios::out
    - Файлът се разрушава и put-указателят се установява в началото. Следващите операции за вмъкване се реализират в текущата позиция на put-указателя
  - ios::app
    - Файлът не се разрушава
    - Не е възможно вмъкване на произволни места във файла
- След вмъкване put-указателят се премества след вмъкнатото
- Използват се същите начини за писане като при cout – operator<< и т.н.
- Пример

Отваряне на текстов файл както за четене, така и за писане

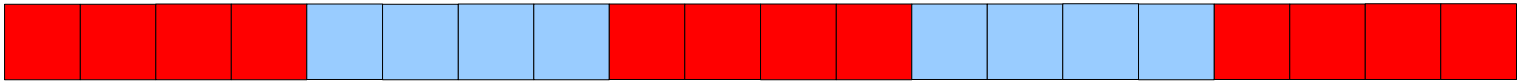
- Конструиране на fstream обект
- Файлът не се разрушава
- Операциите за извличане и вмъкване са възможни на произволни места във файла
- Активират се put и get-указатели, които се установяват в началото на файла
- След всяка операция по четене/писане и двата указателя се преместват след извлеченото или вмъкнатото съответно
- ios::in | ios::ate
  - put и get указателите са в края на файла
  - Опитите за извличане след подходящо позициониране на get указателя са успешни, но опитите за вмъкване са неуспешни
- Пример

Двоични файлове

- Добавя се ios::binary към режимите
- Използват се методи read и write вместо operator<<, operator>>, getline и т.н.
- `istream& read(char* var_str, streamsize size);`
  - Метод на istream
  - var_str е масив от символи, в който се записва прочетеното
  - var_str не се разглежда като символен низ!
  - Четат се най-много size на брой байта
- `ostream& write(const char* str, streamsize size);`
  - Метод на ostream
  - Записва size на брой символи от str

- `istream& seekg(streampos, seekdir = beg)`
- `ostream& seekp(streampos, seekdir = beg)`
- `streampos tellg() const`
- `streampos tellp() const`
- `enum seekdir { beg, cur, end };`

Запис/четене на структури в/от двоичен файл



```
class Student { ... };

Student s;
f.seekp(i * sizeof(Student));
f.write((char const*)&s, sizeof(Student));

Student sa[3];
f.seekg(j * sizeof(Student));
f.read((char*)sa, 3 * sizeof(Student));
```

Обобщение