Потоци и файлове

ios, istream, ostream, iostream, ifstream, ofstream, fstream

Член функции за входно/изходни операции

Както останалите класове, така и тези си имат вградени функции. Сега най-накрая ще се разбере как всъщост работят всички функции, които сте използвали наготово със cin, cout и т.н.

Функции за изходни потоци:

Методи на класа *ostream*

- `put`
  - `ostream& put(char __c);`, където `__c` е символ;
  - Записва символа, който е стойност на аргумента в изходния поток, чрез който член-функцията е активирана.(Активирана, тоест cout.put('f')
  - Тъй като функцията връща референция към потока *ostream*, можем да ползваме *каскадно обръщение* (следната конструкция на писане) :

    ```
    cout.put('c').put('b').put('\n');
    ```

  Пример:

  ```
  cout.put('c');//извежда на екрана символа 'c'
  ```

- `write`
  - `ostream& write(const char* __s, streamsize __n);`, където:
    - `__s` е символен низ;
    - `__n` е целочислен израз, който указва броя на символите за записване в стрийма;
  - Най-често се използва при работа с файлове с пряк или произволен достъп.(по-надолу се разбира какво значи)
  - Записва `__n` на брой последователни символа от низа `__s` в изходния поток, чрез който член функцията е активирана.
    - Ако стойността на `__n` е по-малка от дължината на `__s` се осъществава записване на символите от `__s` до достигане броя на `__n`.
    - Ако стойността на `__n` е по-голяма от дължината на `__s` се осъществава допълване с произволни знажи до размер на полето, равен на `__n`.

  Пример:

  ```
  cout.write("something", 9);//извежда на екрана "something"
  cout.write("something", 4);//извежда на екрана "some"
  cout.write("something", 24);//ще изведе "something", а другите символи
  до допълване на броя 24 ще ги вземе буферите, с които работят
  стриймовете. Понякога може да няма нищо в буфера и ще си изпринти само
  "something", това обаче не значи, че работи безопасно
  ```

  Ако го направим да пише във файл, ще пише допълнителни символи

```
ofstream file;
file.open("/home/zspopov/OOP/Practice/Streams/text.txt", ios::out);

if(file.is_open()){
    file.write("something", 24);
}

file.close();
```

- Функцията разглежда терминиращата нула `'\0'` като обикновен символ, докато оператор `<<` я счита за специален

```
cout << "Something\0is\0going\on!";//извежда "Something"
cout.write("Something\0is\0going\0on!", 15);//извежда "Somethingisgo"
```

Функции на входни потоци:

Методи на класа *istream* и *ifstream*:

- `get` - функцията им 5 overload-а, ще разгледам някои от тях:
  - `istream& get(char& __c);`, където `__c` е референция към char променлива, в която ще записваме входящия символ
    - Извлича един символ от входния стрийм, чрез който е активирана, и го свързва с променливата, указана като нейн аргумент.

```
char ch1, ch2, ch3;
cin.get(ch1).get(ch2).get(ch3);
```

    - За разлика от оператор `>>` функцията може да се използва за въвеждане на интервали `' '`, табулация `'\t'` и нов ред `'\n'`

  Пример: Ако работим с потоци, които са насочени към файлове можем

```
char ch;
while(fin.get(ch)){
    fout.put(ch);
}
```

  да направим дословно копие на файла, свързан с fin, във файла, вързан с изходния поток fout, докато чрез оператора `>>`

```
while(fin >> ch){
    fout << ch;
}
```

  интервалите, знаците за табулация и за преминаване на нов ред няма да бъдат прекопирани

    - Често се използва заедно с *while* цикъл, като можем да означим символ, който да направим проверка за въведен символ

  Пример:

```
char ch = '\0';
while(ch != 'q'){
    cin.get(ch);
    //и вече си работите със символите искате да ползвате символа
}
```

- `istream& get(char * var_str, streamsize size);`

  `istream& get(chst * var_str, streamsize size, char delim);`,

  - където:
    - `var_str` е променлива от тип указател към символ, представяща масив от символи, в която се записва извлеченият чрез *get* символен низ;
    - `size` е целочислен израз, означаващ максималния брой извлечени символи;
    - `delim` (от delimeter (разделител)), ако е пропуснат се подразбира `'\n'`;
  - Извлича, ако е възможно *size - 1* последователни символи от входния поток, чрез който е активирана, и ги запомня като низ в променливата *var_str*, указана като първи аргумент. Ако при извличането е достигнат указаният в третия параметър разделител, извличането спира до символа преди разделителя. Извличането също спира при изчерпване на символите в потока (например при достигане на края на някой файл) или при възникване на грешка по време на операция. Ако извличането е продължило до достигане на разпределителя, разприделителят не се записва в *var_str*,