ФУНКЦИИ СТАНДАРТНОЙ БИБЛИОТЕКИ

Функции стандартной библиотеки

- Функции ввода/вывода.
- Функции работы со строками и символами.
- Математические функции.

ФУНКЦИИ СТАНДАРТНОЙ БИБЛИОТЕКИ.

Для использования в программе обращения к стандартной библиотеке, нужно с помощью директивы **#include** *включить в исходный текст программы заголовочные файлы, в которых находятся соответствующие объявления.*

- *Сами библиотечные функции хранятся в скомпилированном виде и подключаются к программе на этапе компоновки.*

В программах на **C++** *могут использоваться функции, унаследованные от библиотеки* **C**.

Функции ввода/вывода

- Ввод/вывод в C++ реализуется либо с помощью функций, унаследованных от библиотеки C, либо с помощью потоков C++.
- Смешивать эти два способа в одной программе можно только синхронизировав ввод с помощью функции **sync_with_stdio()**.

ФУНКЦИИ ВВОДА/ВЫВОДА.

Каждый способ имеет свои преимущества:

- *Преимущество использования потоков в том, что они легче в использовании в простых случаях ввода/вывода, не требующих форматирования, а, главное, потоковые операции можно переопределить для собственных классов.*
- *Ввод/вывод в стиле* **C** *удобнее использовать при форматированном выводе в программах, не использующих объектно-ориентированную технику.*

Кроме того, существует множество программ, написанных на **C** *и перенесенных на* **C++**, *с которыми программисту приходится сталкиваться.*

ФУНКЦИИ ВВОДА/ВЫВОДА.

Для использования функций ввода/вывода в стиле C необходимо подключить к программе заголовочный файл **<stdio.h>** *или* **<cstdio>**.

При вводе/выводе данные рассматриваются как

Физически поток представляет собой файл или устройство (например, клавиатуру или дисплей).

ОТКРЫТИЕ ПОТОКА.

Работа с потоком начинается с его открытия.

Поток можно открыть для чтения или записи в двоичном или текстовом режиме.

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

- *Первый параметр — имя открытого файла в виде С-строки*
- *Второй параметр — режим открытия файла*

При успешном открытии потока функция возвращает указатель на

режим открытия файла:

mode	действия
"r"	*файл открывается для чтения*
"w"	*открывается пустой файл для записи (если файл существует, **он стирается**)*
"a"	*файл открывается для добавления информации в его конец*
"r+"	*файл открывается для чтения и записи (файл должен существовать)*
"w+"	*открывается пустой файл для чтения и записи (если файл существует, **он стирается**)*
"a+"	*файл открывается для чтения и добавления информации в его конец*
"t"	*текстовый режим*
"b"	*двоичный режим*

Открытие потока.

ОТКРЫТИЕ ПОТОКА.

*По умолчанию файл открывается в текстовом режиме, при котором комбинация символов «возврат каретки» и «перевод строки» (**0x13 0x10**) при вводе преобразуются в одиночный символ перевода строки.*

- *при выводе выполняется обратное преобразование*

В двоичном режиме эти преобразования не выполняются.

Пример:

```
FILE *f = fopen("d:\\cpp\\data", "rb+");
```

Указатель **f** *используется в дальнейших операциях с потоком.*
Его передают функциям ввода/вывода в качестве параметра.

Открытие потока.

Открытие потока.

При открытии потока с ним связывается область памяти, называемая *буфером.*

- *При выводе вся информация направляется в буфер и накапливается там до заполнения буфера или до закрытия потока.*

Чтение осуществляется блоками, равными размеру буфера, и данные читаются из буфера.

- *С помощью функций* **setbuf** *и* **setvbuf** *можно управлять размерами и наличием буферов.*

Следует иметь в виду, что при аварийном завершении программы выходной буфер может быть не выгружен, и возможна потеря данных.

- *Существует пять предопределенных потоков, которые открываются в начале работы программы.*
- 1.стандартный ввод **stdin**
- 2.стандартный вывод **stdout**
- 3.стандартный вывод сообщений об ошибках **stderr**
- 4.стандартный дополнительный поток **stdaux**
- 5.стандартная печать **stdprn**

Первые три потока по умолчанию относятся к консоли.

Эти указатели можно использовать в любой функции ввода/вывода там, где требуется указатель потока.

Открытие потока.

ВВОД/ВЫВОД В ПОТОК.

Ввод/вывод в поток можно осуществлять различными способами:

- *в виде последовательности байтов*
- *в виде символов и строк*

Для каждого вида операций определен свой набор функций.

Операции ввода/вывода выполняются, начиная с текущей позиции потока, определяемой положением ***указателя потока.***

Указатель устанавливается при открытии на начало или конец файла (*в соответствии с режимом открытия*) и изменяется автоматически после каждой операции ввода/вывода.

Текущее положение указателя можно получить с помощью функций **ftell** и **fgetpos** и задать явным образом с помощью функций **fseek** и **fsetpos**.

- Эти функции нельзя использовать для стандартных потоков.

Ввод/вывод в поток.

Основные функции ввода/вывода потока :

действие	функции
чтение потока байтов	**fread**
запись потока байтов	**fwrite**
чтение символа из потока	**getc, fgetc**
чтение символа из стандартного потока **stdin**	**getchar**
запись символа в поток	**putc, fputc**
запись символа в стандартный поток **stdout**	**putchar**
чтение строки из потока	**fgets**
чтение строки из стандартного потока **stdin**	**gets**
запись строки в поток	**fputs**
запись строки в стандартный поток **stdout**	**puts**

Ввод/вывод в поток.

Основные функции ввода/вывода потока :

действие	функции
форматированный ввод из потока	**fscanf**
форматированный ввод из стандартного потока **stdin**	**scanf**
форматированный ввод из строки	**sscanf**
форматированный вывод в поток	**fprintf**
форматированный вывод в стандартный поток **stdout**	**printf**
форматированный вывод в строку	**sprintf**

Ввод/вывод в поток.

ЗАКРЫТИЕ ПОТОКА.

Поток закрывается либо при завершении программы, либо явным образом с помощью функции **fclose**:

int fclose(FILE*);

Перед закрытием потока информация из связанных с ним буферов выгружается на диск.

Рекомендуется всегда явным образом закрывать потоки, открытые для записи, чтобы избежать потери данных.

ОБРАБОТКА ОШИБОК.

Функции работы с потоком возвращают значения, которые рекомендуется анализировать в программе и обрабатывать ошибочные ситуации.

- *например, при открытии существующих файлов или чтении из потока.*

***При работе с файлами часто используются функции* feof *и* ferror.**

	•*возвращает не равное нулю значение, если достигнут конец файла, в противном случае 0*
	•*возвращает не равное нулю значение, если обнаружена ошибка ввода/вывода, в противном случае 0*

Обработка ошибок.

Постановка задачи

- *В файле хранятся сведения о мониторах:*
  - ***тип, оптовая и розничная цены и примечание.***
- *Данные в каждой строке записаны единообразно:*
  - *20 символов — тип монитора;*
  - *5 символов — целое число, представляющее оптовую цену;*
  - *5 символов — целое число, представляющее и розничную цену;*
  - *40 символов — примечание.*
- *Необходимо:*
  - *построчно считать данные из текстового файла в буферную переменную* **s**;
  - *сформировать из них структуру* **mon**;
  - *записать* **mon** *в двоичном режиме в выходной файл;*
  - *считать из этого файла произвольную запись.*

Пример работы с потоками.

```
#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(){
      FILE *fi, *fo;
      if((fi = fopen("d:\\temp\\file.txt", "r")) == 0){
            cout << " Ошибка открытия входного файла "; return 1;}
      if((fo = fopen("d:\\temp\\binfile.out", "w+b")) == 0){
            cout << " Ошибка открытия выходного файла "; return 1;}
      const int dl = 80;
      char   s[dl];
      struct{
            char type[20];
            int  opt, rozn;
            char comm[40];
      }mon;
      int kol = 0;        // Количество записей в файле
```

Пример работы с потоками.

```
while (fgets(s, dl, fi)){
            // Преобразование строки в структуру:
        strncpy(mon.type, s, 19);
        mon.type[19]='\0';
        mon.opt = atoi(&s[20]);     // Описание atoi см. в след. разделе
        mon.rozn = atoi(&s[25]);
        strncpy(mon.comm, &s[30], 40);
        fwrite(&mon, sizeof mon, 1, fo);
        kol++;
}
fclose(fi);
```

Пример работы с потоками.

```
cout << " Введите номер записи:";
int i; cin >> i;    // Номер записи
      if (i >= kol){cout << " Запись не существует "; return 1;}
      // Установка указателя текущей позиции файла на запись i:
      fseek(fo, (sizeof mon)*i, SEEK_SET);
      fread(&mon, sizeof mon, 1, fo);
      cout      << "mon.type " << mon.type << " opt " << mon.opt
            << " rozn " <<  mon.rozn << endl;
       fclose(fo);
       return 0;
}
```

Пример работы с потоками.

Функции работы со строками и символами

- Строка представляет собой массив символов, заканчивающийся нуль-символом.

ФУНКЦИИ РАБОТЫ СО СТРОКАМИ И СИМВОЛАМИ.

В C++ есть две возможности работы со строками:

- *функции, унаследованные из библиотеки* **C** *(заголовочный файл* **<string.h>** *или* **<cstring>***).*
- *библиотечный класс* **C++ string**, *предоставляющий более широкие возможности представления, обработки и контроля строк.*

- Библиотека **C** содержит функции :
  - копирования строк (**strcpy, strncpy**)
  - сравнения строк (**strcmp, strncmp**)
  - объединения строк (**strcat, strncat**)
  - поиска подстроки (**strstr**)
  - поиска вхождения символа (**strchr, strrchr, strpbrk**)
  - определения длины строки (**strlen**)
  - и другие

Функции работы со строками.

полезные функции преобразования строк в числа (файлы **<stdlib.h>** и **<cstdlib>**):

функция	действия
double atof(const char* p)	*преобразует переданную строку в* **double**
int atoi (const char* p)	*преобразует переданную строку в* **int**
long atol(const char* p)	*преобразует переданную строку в* **long**

- *Пробелы и табуляции в начале строки пропускаются.*
- *Преобразование прекращается при встрече недопустимого символа или конца строки.*
- *Если строку нельзя преобразовать в число, возвращается* **0**.
- *Если число выходит за пределы диапазона данного типа, переменной* **errno** *(заголовочный файл* **<cerrno>***) присваивается значение* **ERANGE** *и возвращается допустимое число.*

обратные преобразования можно сделать с помощью функции **sprintf**

Функции работы со строками.

Пример:

(программа заполняет массив типа **double** из строки)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main(){
     char s[] = "2, 38.5, 70, 0, 0, 1", *p = s;
     double m[10];
     int i = 0;
     do{
         m[i++] = atof(p);
         if (i>9) break;
     } while(p = strchr(p, ','), p++);
     for( int k = 0; k<i; k++) printf("%5.2f ", m[k]);
     return 0;
}
```

функции проверки на принадлежность символа множеству (файлы **<ctype.h>** и **<cctype>**):

Имя	Проверка на принадлежность символа множеству
isalnum	*букв и цифр* **(A-Z, a-z, 0-9)**
isalfa	*букв* **(A-Z, a-z)**
iscntrl	*управляющих символов* **(с кодами 0..31 и 127)**
isdigit	*цифр* **(0-9)**
isgraph	*печатаемых символов, кроме пробела* **(isalfa | isdigit | ispunct)**
islower	*букв нижнего регистра* **(a-z)**
isprint	*печатаемых символов*
ispunct	*знаков пунктуации*
isspace	*символов-разделителей*
isupper	*букв верхнего регистра* **(A-Z)**
isxdigit	*шестнадцатеричных цифр* **(A-F, a-f, 0-9)**

Функции принимают величину типа **int** *и возвращают значение* **true**, *если условие выполняется*

Для каждой из перечисленных функций есть ее аналог для многобайтных символов типа **wchar_t**, *содержащий в названии букву* **w**

Функции работы с символами.

Математические функции

- Строка представляет собой массив символов, заканчивающийся нуль-символом.

МАТЕМАТИЧЕСКИЕ ФУНКЦИИ.

C++ *унаследовал из* **C** *стандартные математические функции, описание которых находится в заголовочных файлах* **<math.h>** (**<cmath>**).

Они позволяют получить:

- *абсолютное значение* (**abs, fabs**)
- *округленное число* (**ceil, floor**)
- *квадратный корень* (**sqrt**)
- *степень* (**pow**)
- *значения тригонометрических функций* (**sin, cos, tan,**