Программирование и алгоритмизация

Санкт-Петербургский государственный политехнический университет

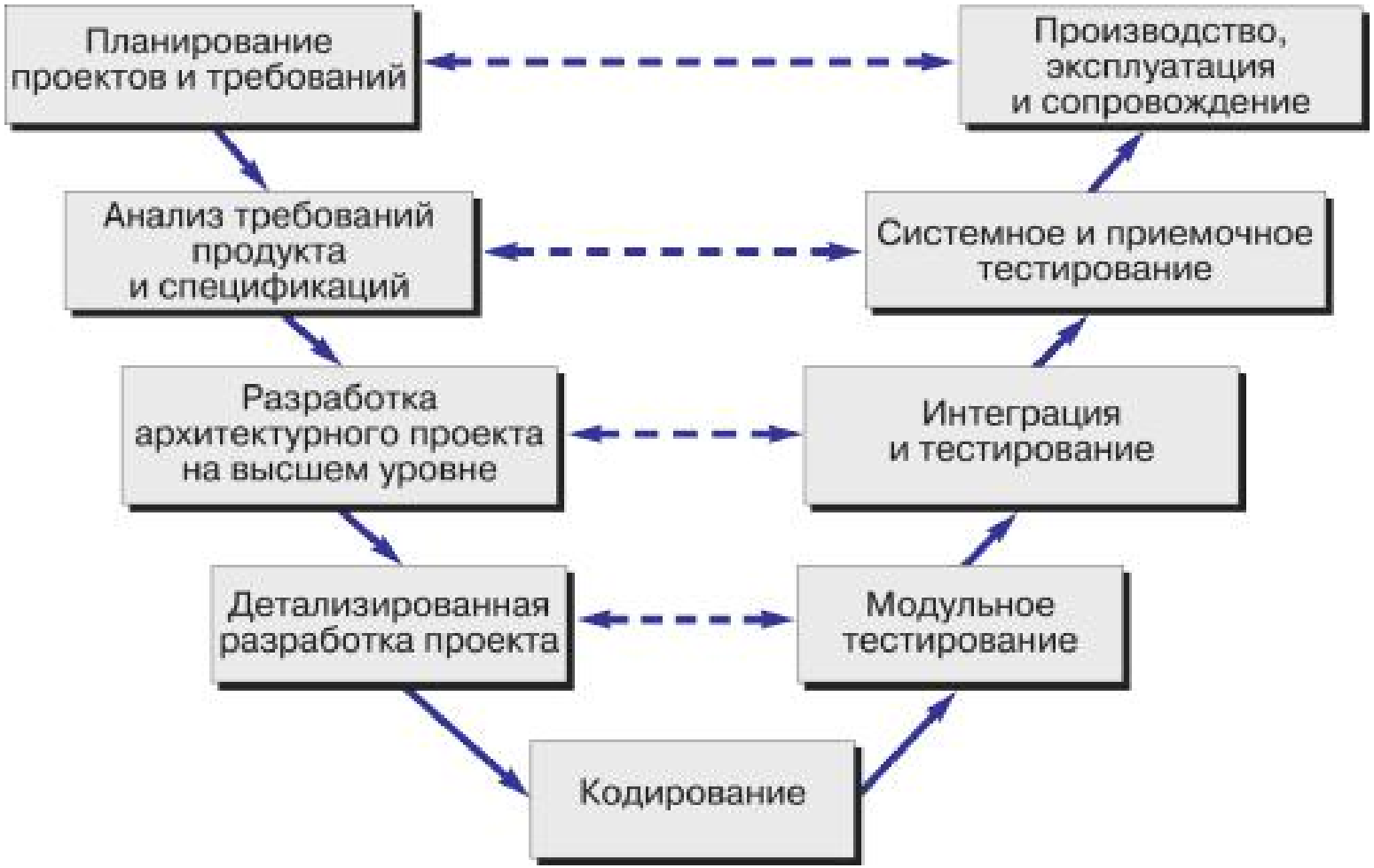
04 сентября 2014

Процесс разработки

Существует множество моделей процесса разработки программного обеспечения. Одной из таких моделей является «Водопадная (Каскадная)» модель. В этой модели процесс разработки выглядит как поток, последовательно проходящий фазы:

1. Определение требования
2. Проектирование
3. Кодирование
4. Интеграция
5. Тестирование и отладка
6. Инсталляция
7. Поддержка

V-образная модель процесса разработки



Критерии качества программы

Структура программы влияет на:

- скорость разработки (time-to-market);
- надежность;
- возможность легко модифицировать;
- возможность легко сопровождать;

Количество потребляемых ресурсов программой зависит от:

- эффективность алгоритмов (вычислительная сложность);
- объем занимаемой памяти;

Структура программы

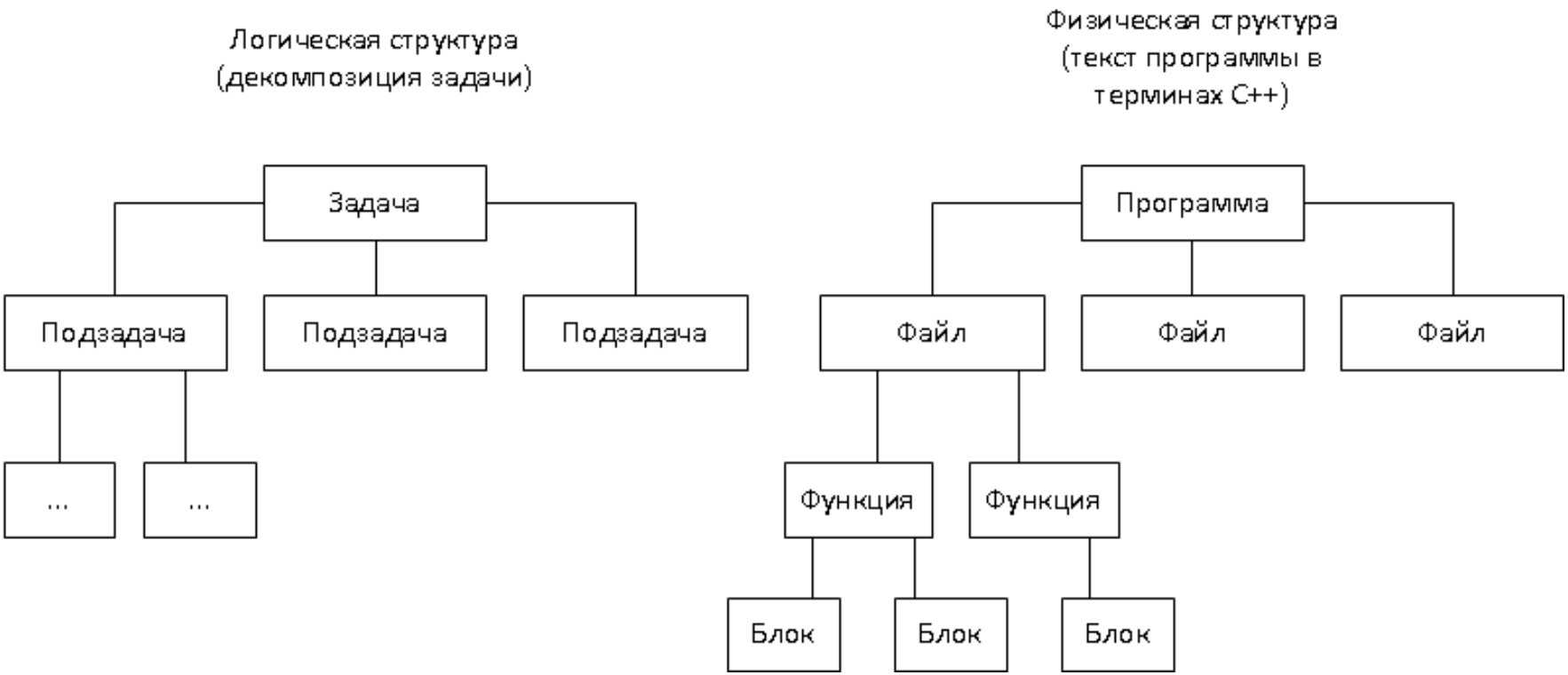
Б. Страуструп: «Вы можете написать небольшую программу (скажем, не более 1000 строк), используя грубую силу и нарушая все правила хорошего стиля программирования. Если структура программы, состоящей из 100 000 строк, плоха, Вы обнаружите, что новые ошибки появляются с той же скоростью, с которой исправляются старые. C++ создавался с целью, чтобы большую программу можно было структурировать таким образом, чтобы один человек мог работать с большим объемом кода».

Структурное программирование

Это методология и технология разработки ПО, основанная на принципах программирования «сверху-вниз».
При этом программист создает текст программы, который:

- отражает структуру решаемой задачи (логическую структуру);
- хорошо читаем не только его создателем, но также и другими программистами.

Структурное программирование



Эволюция языков C и C++

- 1969–1973 — Первые версии языка Си (как развитие языка Би).
- 1989 — Принят стандарт C89.
- 1999 — Стандарт языка C99.
- 1979 — Расширение языка Си (Си с классами).
- 1983 — Расширенный язык переименован в C++.
- 1985 — В C++ добавлено закрытое наследование, шаблоны и т.д.
- 1998 — Стандарт языка C++ "Standart for the C++ Programming Language".
- 2011 — Новый стандарт языка C++11x(C++0x).

Разбиение на файлы (модульность)

Разбиение программы на файлы помогает:

- улучшить структуру программы;
- при грамотном разбиении уменьшить общее время получения нового исполняемого модуля

Для обеспечения модульности C++:

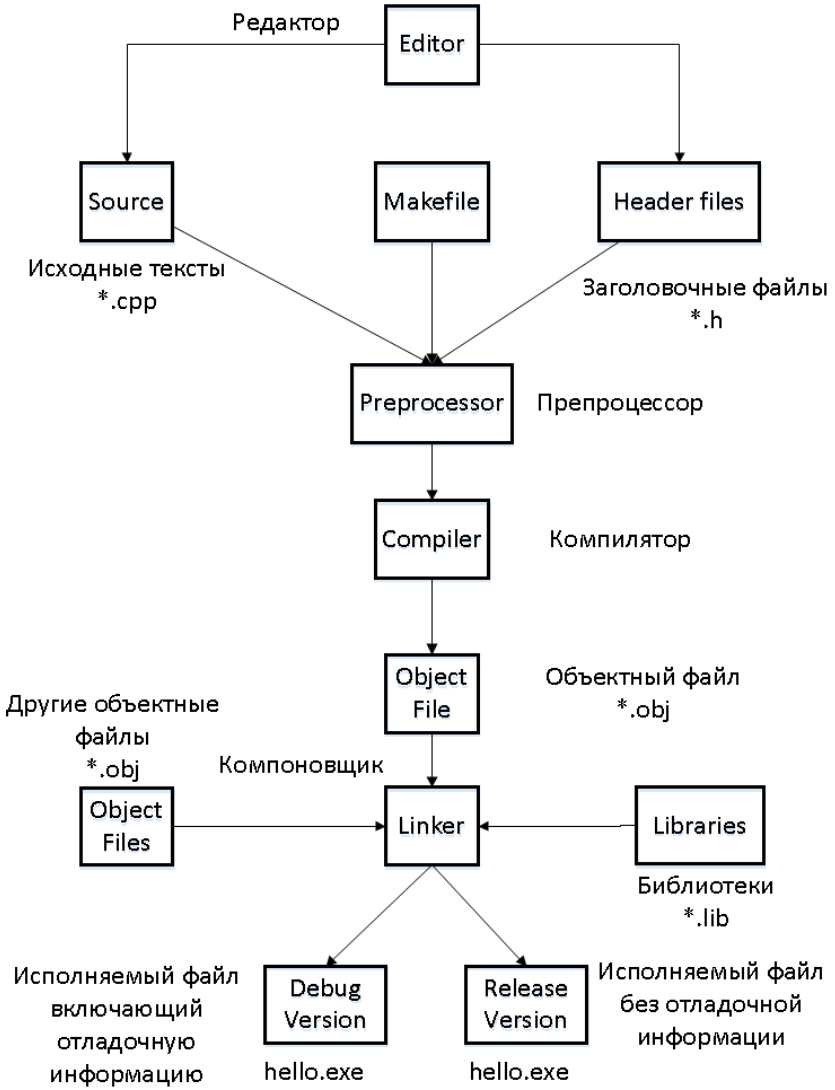
- предоставляет возможность компиляции каждого файла по отдельности;
- последующую стыковку полученных частей в единый исполняемый модуль (приложение).

Реализация языка C/C++

Включает в себя:

- Компилятор — набор ключевых слов и конструкций.
- C/C++ Runtime library — стандартная библиотека готовых функций поставляемая с компилятором.

Последовательность этапов генерации исполняемого кода



Препроцессор

Это текстовый редактор, который автоматически запускается перед этапом компиляции.
Результатом работы препроцессора является файл с иходным текстом программы, с которым уже может работать компилятор. Такой файл называется единицей компиляции.

Компиляция

Включает в себя:

- лексический анализ;
- синтаксический анализ;
- трансляция;

Результатом является промежуточный объектный файл, обычно имеющий расширение .obj или .o Ошибки этапа компиляции: синтаксические — как правило легко находятся, компилятор указывает строки с ошибками.

Компоновка

Этап соединения всех ранее откомпилированных частей (объектных файлов). На данном этапе все объектные модули (статические и динамические) обрабатываются одновременно для распределения памяти и формирования адресных частей для всех команд. Ошибки этапа компоновки: неразрешенные или неуникальные внешние зависимости. Такие ошибки находить сложнее, т.к. не указан конкретный файл с исходным текстом где произошла ошибка.

Логические ошибки

Обнаруживаются сложнее всего, обычно содержатся в алгоритмах, либо синтаксических конструкциях, которые компилятор понимает иначе, чем программист.

Специфика языка C++

- Совместим с языком C;
- Эффективность:
  - Обладает низкоуровневыми возможностями.
  - Нет сборщиков "мусора".
- Синтаксически сложный язык (выучить не возможно).

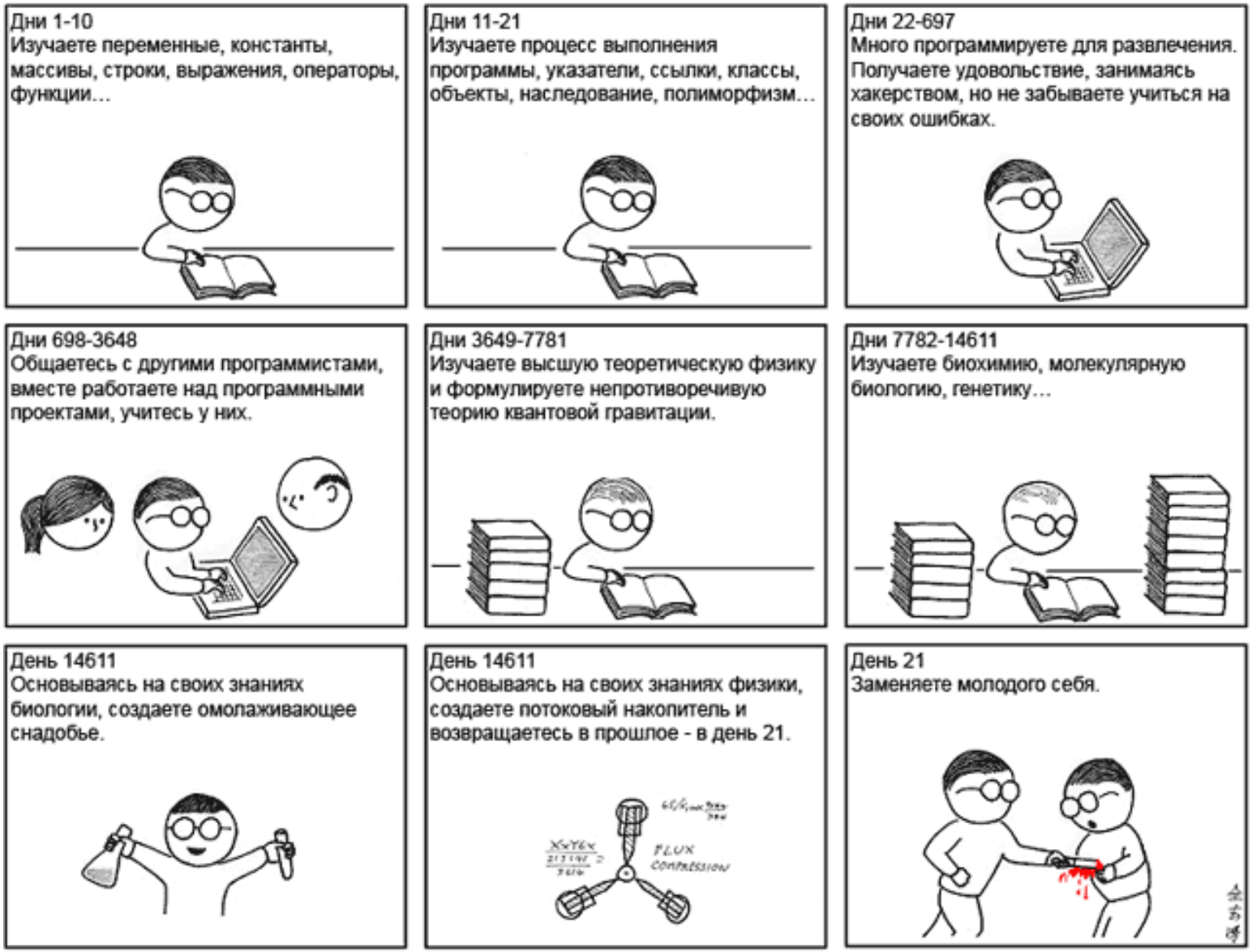
C/C++ язык конструкций

- int x;
- int *x;
- int* x;
- int * x;
- int x[10];
- int* x[10];
- int* (*x)(int);

Литература

1. М. И. Полубенцева. C/C++. Процедурное программирование. БХВ-Петербург 2008.
2. Т. А. Павловская. C/C++ Программирование на языке высокого уровня. Питер 2004.
3. Б. В. Керниган, Д. М. Ричи. Язык С.
4. Х. М. Дейтел, П. Дж. Дейтел. Как программировать на C++. Бином 2004.

C++ за 21 день



Насколько мне известно, это самый легкий способ «Выучить C++ за 21 день».

Понятие проекта

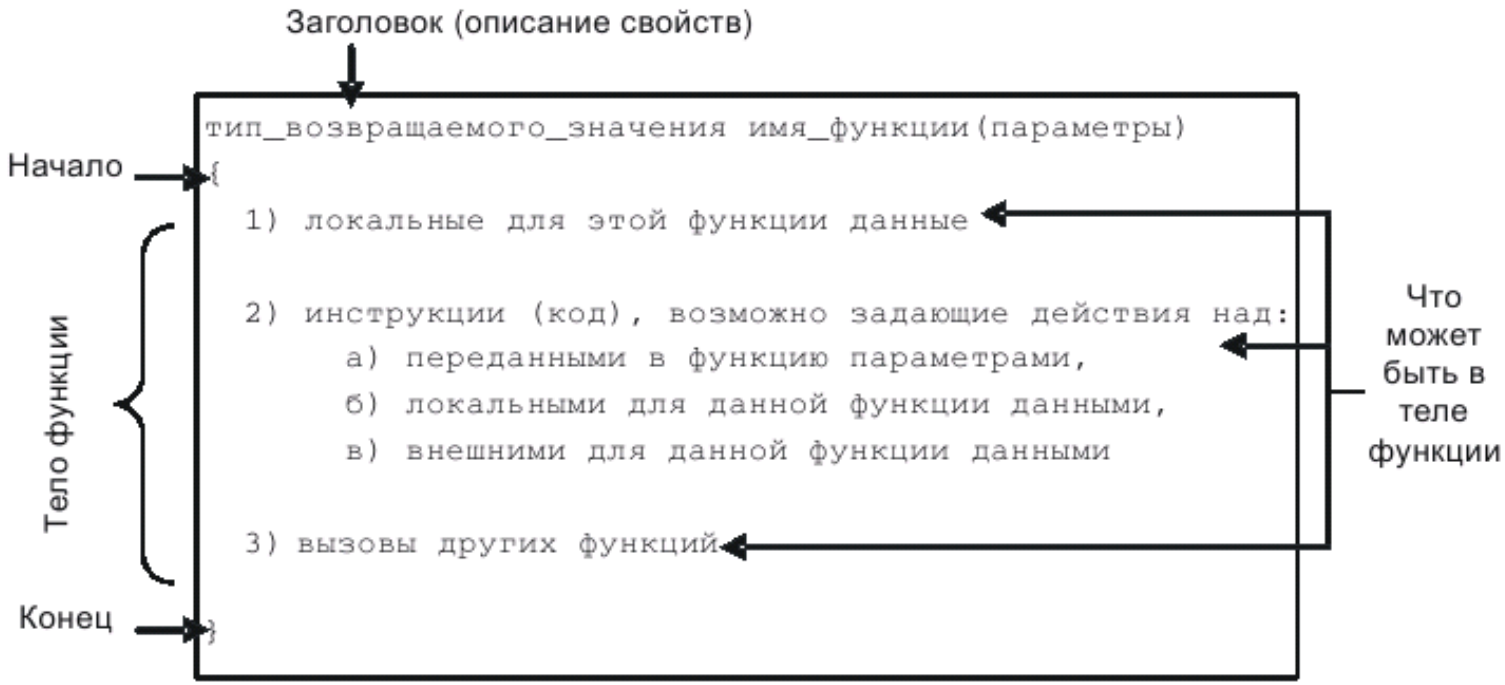
Совокупность:

- файлов с исходными текстами программы. Обучно имеют расширения: .c, .cpp, .cc, .cxx, .h .
- служебных файлов среды разработки, в них содержится вспомогательная информация, например, базы автодополнения IntelliSense.

Каждому проекту соответствует исполняемый файл: .exe. Либо библиотека: .dll, .lib, .a.

Функции

Это средство, позволяющее выполнять один и тот же код с разными наборами данных.
При структурном программировании, с использованием функций: улучшается структура(читаемость) текста программы; позволяют избежать дублирования кода.



Функция main

В теле программы на C/C++, всегда должна быть описана одна функция с именем main. Эта функция не генерируется автоматически и не предоставляется стандартной библиотекой. !!! Наличие этой функции обязательно !!! .

Комментарии

Это часть документирования программы. Пишутся для программистов — компилятору они не нужны.
Удачно подобранный и грамотно написанный набор комментариев является неотъемлемой частью хорошей программы.

```
// однострочный комментарий первого вида, в стиле C
   ++ (или C99).
/* многострочный комментарий
   второго вида в стиле C
*/
/*
 //так можно вкладывать комментарии
*/

/*
/*
  а так нельзя :(
*/
*/
```

Рекомендации по документированию кода

- плохой комментарий хуже чем его отсутствие (Б.Страуструп);
- не стоит комментировать то, что и так очевидно;
- хорошим тоном считается начинать с комментария каждый файл и определение каждой функции;
- всегда комментируйте непереносимый и сложный код;
- обязательно комментируйте понятия, которые используются разными единицами трансляции (особенно если ими может воспользоваться другой прогаммист);
- для сопровождения программы хорошо написанный комментарий ЦЕННЕЕ самого кода.

Оформление блоков кода

```
{ //начало блока
 { //вложенный блок
   { //еще
   } //вложенный блок
 } //конец вложенного блока
} //конец блока
```

Ключевые слова и понятия

Ключевые слова зарезервированные компилятором применять можно только по назначению. Сюда относятся: if, int, static, struct, long, unsigned, ...

Стандартные типы данных в C++

- char — символ;
- int — целое число;
- float — число с плавающей точкой;
- double — число с плавающей точкой двойной точности;
- enum — перечислимый тип задает набор именованных констант целого типа.

Идентификаторы

Идентификаторы используются для задания имен переменных, функций, констант и т.п.

- в качестве идентификаторов нельзя использовать ключевые слова C/C++;
- компилятор различает верхний и нижний регистр букв. Hello и hello — две разные переменные;
- чем шире область использования тем осмысленее должны быть имена;
- первым символом должна быть буква или символ подчеркивания _;
- идентификатор состоит из последовательности букв и цифр.

Стандартные типы данных в C++

Уточняющие описатели типов: short, long и unsigned делают возможными следующие описания типов:

- unsigned char
- short int
- long int
- unsigned short int
- unsigned int
- unsigned long int

lvalue и rvalue

lvalue — конструкция, которая может быть использована слева от знака равенства. x=1; //переменная x является lvalue.
В общем случае lvalue — любое выражение, результат которого компилятор ассоциирует с выделенной памятью. В качестве lvalue, может быть переменная, либо сложная конструкция, либо справа тоже может стоять lvalue:

- x=<выражение>;
- *f()=<выражение>;
- x=y=<выражение>;

НО нельзя строить выражения:

- x+y=<выражение>;
- 10=<выражение>

rvalue — все что может находиться справа от знака равенства, например, константы.

Область действия имен

```
int i=16; // Глобальная переменная
void proc()
{
  int i=1;
  {
    int i=2;
    printf ("\n global i=%d, local i=%d",::i, i);
  }
}
```

Операции языка C++

Оператор означает действие, которое может быть совершено над одним и более операндами.
3 типа операций:

- унарные
- бинарные
- тернарные

Унарные

$- \sim !$	Операции отрицания и дополнения
* &	Разыменование и взятие адреса
sizeof	Определение размера переменной или типа
+	Унарный плюс
++ – –	Приращение (increment) и убывание (decrement)

Бинарные

* / %	Мультипликативные операции
+ −	Аддитивные операции
<< >>	Операции сдвига
< > <= >= == !=	Операции бинарных отношений
& | ^	Побитовые логические операции
&& ||	Логические операции
,	Операции последовательного вычисления

Приоритеты операций

Операторы включающие в себя присваивание раскрываюся справа налево. Тернарные операторы также являются правоассоциативными.
Остальные операторы такие как: сложение, умножение, сравнение являются левоассоциативными.

- a=b=c, эквивалентно a=(b=c).
- a+b+c, эквивалентно (a+b)+c.