C++

C++ — компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности.

C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. Наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Область применения языка включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр).

Основные достоинства языка – компактный синтаксис, наличие большого количества специальных средств, упрощающих написание сложных системных программ, многоплатформенность.

Основной недостаток – «незащищенный» синтаксис, при котором в языке возможно существование близких по форме допустимых конструкций, что часто не позволяет идентифицировать ошибку на этапе компиляции программы, а потому удлиняет и усложняет ее отладку.

Широко распространены следующие компиляторы C/C++:

gcc – GNU C Compiler;

Microsoft Visual C++;

Intel C++ Compiler.

Компилятор gcc (GNU C Compiler) является свободно распространяемым программным продуктом и является де-факто стандартом для сборки C/C++ программ под операционными системами Linux и FreeBSD. Реализован для множества аппаратных платформ и различных операционных систем. Реализация для ОС Windows называется mingw.

Используется для компиляции свободно распространяемых программных продуктов с открытыми

Коммерческий компилятор Microsoft Visual C++ для процессоров семейств x86, x86-64 и IA-64 наиболее распространенный компилятор для создания приложений для ОС Windows (включая различные её версии для различных платформ Win32, Win64, WinCE).

Чаще всего используется совместно со средой разработки MS Visual Studio. В отличии от gcc, ориентируется не на соблюдение принятых стандартов C/C++ как таковых, а на внутренние спецификации Microsoft. Компилятор лучше оптимизирует код программ, чем mingw или gcc, однако не полностью с ним совместим.

Коммерческий компилятор Intel C++ Compiler для процессоров семейств x86, x86-64 и IA-64 позиционируется как оптимизирующий компилятор для приложений, критических к скорости работы или аппаратным ресурсам. Может использоваться в качестве замены Microsoft Visual C++ для ОС MS Windows, в том числе совместно с MS Visual Studio.

Данный компилятор также существует для ОС Linux и Mac OS, однако распространен мало и используется только для создания коммерческих приложений. Не полностью совместим с gcc.

ЭТАПЫ РЕШЕНИЯ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ

Программа – это запись алгоритма на языке программировании. Ее можно считать законченным результатом процесса разработки алгоритма, готовым к использованию.

1 Постановка задачи. Разбираемся с условием задачи, что дано, что нужно найти.

2 Математическое описание задачи. Построение математической модели.

3 Разработка алгоритма решения задачи. Построение блок-схемы.

4 Написание текста программы в окне редактирования на языке программирования.

5 Компиляция текста в машинный код. Она осуществляется встроенным компилятором среды.

6 Отладка программы. Исправление ошибок, на которые укажет среда.

7 Запуск отлаженной программы и тестирование.

8 Получение и анализ результатов

Тестирование готовой программы бывает двух видов:

- **Открытое –** заключатся в том, что запуск программы осуществляется специально подобранным на основе программного кода. Т.е. данные подбираются с учетом потенциально опасных мест кода.

Основные точки, вызывающие ошибки:

Недостаточная инициализация (забыли присвоить начальное значение или завести переменную).

Неправильное построение логических высказываний (путаница между «и» и «или»).

Не все циклы зациклены (параметризированный выполняется столько раз, сколько указанно в программе, с пред и пост условием от 0 до бесконечности в зависимости от условия, с пред может вообще не выполняться с пост хотя бы 1 раз).

Закрытое – не смотрим в код. Должно проводиться несколько тестов удовлетворяющих требованиям:

1. Должны рассматриваться краевые значения т.е. минимально и максимально возможные.
2. Работа с упорядоченными и перевернутыми наборами данных.
3. Тесты имитирующие интересные случаи. Подобрать данные так, чтобы под выборку попадал 1 уникальный элемент или ни одного.
4. Произвольные случайные тесты.

СТРУКТУРА И ЭТАПЫ СОЗДАНИЯ ПРОГРАММЫ НА ЯЗЫКЕ С++

Программа на языке С++ представляет собой текстовый файл, в котором представлены конструкции и операторы данного языка в заданном программистом порядке. В самом простом случае этот текстовый файл может содержать такую информацию:

```
#include <stdio.h>
int main()
{
printf(“Hello World!”);
return 0;
}
```

и обычно имеет расширение cpp, например, «ex1.cpp».

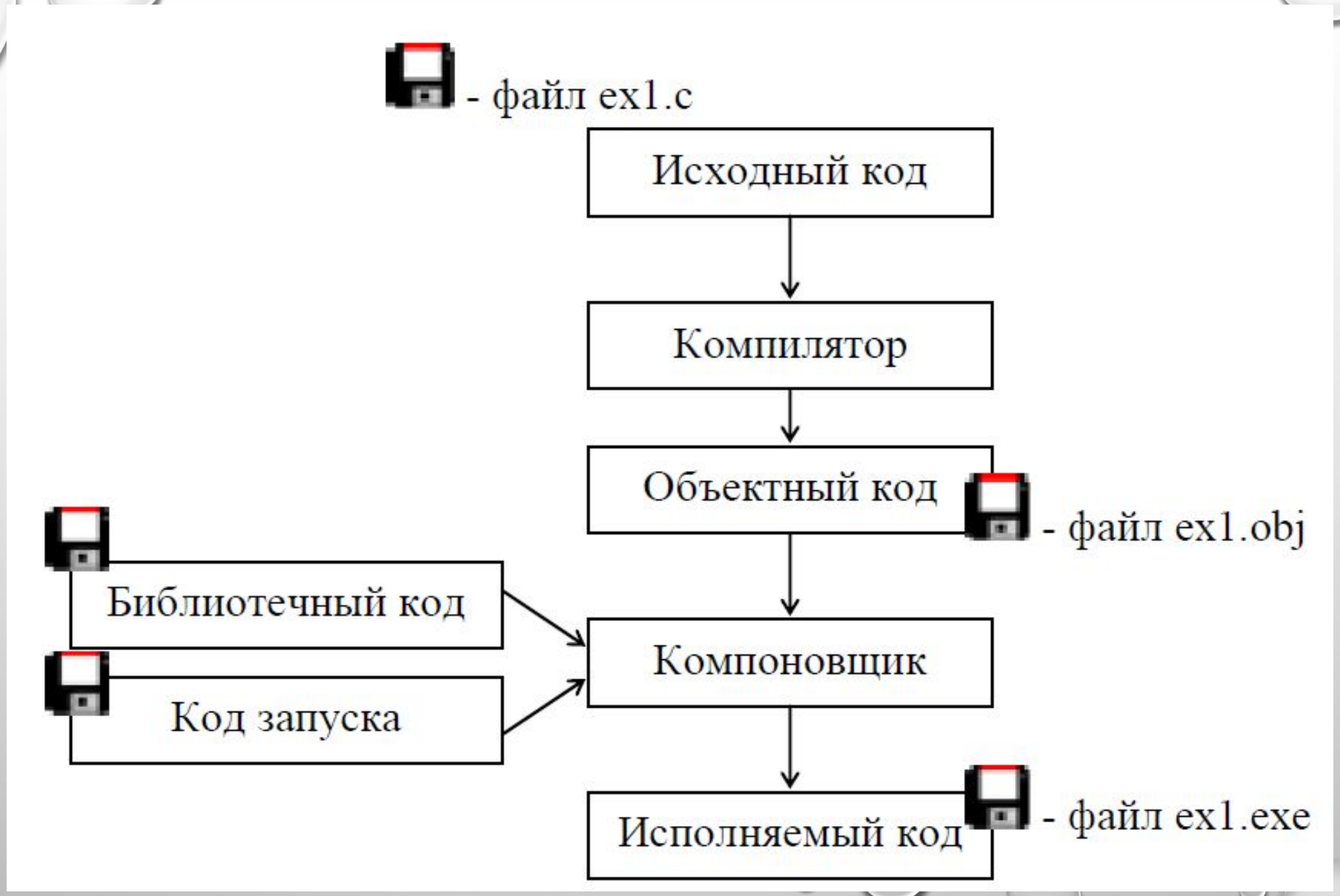
Компиляция – процесс, при котором содержимое текстового файла преобразуется в исполняемый машинный код, понимаемый процессором компьютера. Однако компилятор создает не готовую к исполнению программу, а только объектный код (файл с расширением *.obj). Этот код является промежуточным этапом при создании готовой программы.

Дело в том, что создаваемая программа может содержать функции стандартных библиотек языка C++, реализации которых описаны в объектных файлах библиотек.

Например, в приведенной программе используется функция printf() стандартной библиотеки «stdio.h». Это означает, что объектный файл ex1.obj будет содержать лишь инструкции по вызову данной функции, но код самой функции в нем будет отсутствовать.

Для того чтобы итоговая исполняемая программа содержала все необходимые реализации функций, используется компоновщик объектных кодов.

Компоновщик – это программа, которая объединяет в единый исполняемый файл объектные коды создаваемой программы, объектные коды реализаций библиотечных функций и стандартный код запуска для заданной операционной системы. В итоге и объектный файл, и исполняемый файл состоят из инструкций машинного кода. Однако объектный файл содержит только результат перевода на машинный язык текста программы, созданной программистом, а исполняемый файл – также и машинный код для используемых стандартных библиотечных подпрограмм и для кода запуска.


- файл ex1.c
Исходный код
Компилятор
Объектный код
- файл ex1.obj
Библиотечный код
Код запуска
Компоновщик
Исполняемый код
- файл ex1.exe

Директива #include дает команду препроцессору языка C++ вставить содержимое файла «stdio.h» на место этой строки при компиляции.

Функция main возвращает целое число (тип int) и не принимает никаких аргументов (тип void). Функция main() является обязательной функцией для всех программ на языке C++ и без ее наличия уже на этапе компиляции появляется сообщение об ошибке, указывающее на отсутствие данной функции.

Функция main() является **точкой входа в программу**. В данном случае под точкой входа понимается функция, с которой начинается и которой заканчивается работа программы.

При запуске exe-файла происходит активизация функции main(), выполнение всех операторов, входящих в нее и завершение программы. Таким образом, логика всей программы заключена в этой функции.

В приведенном примере при вызове функции main() происходит вызов функции printf(), которая выводит на экран монитора сообщение «Hello World!», а затем выполняется оператор return, который возвращает нулевое значение. Это число возвращается самой функцией main() операционной системе и означает успешное завершение программы. Фигурные скобки {} служат для определения начала и конца тела функции, т.е. в них содержатся все возможные операторы, которые описывают работу данной функции.

После каждого оператора в языке C++ ставится символ «;».

АЛФАВИТ ЯЗЫКА ПРОГРАММИРОВАНИЯ C++

Программа на языке C++ может содержать следующие символы:

- прописные, строчные латинские буквы A-Z, a-z и знак подчеркивания;
- арабские цифры от 0 до 9;
- специальные знаки: { } , | , [ ] ( ) + / % * . \ : ? < = > ! & # ~ ; ^
- символы пробела, табуляции и перехода на новую строку.

Из символов алфавита формируют ключевые слова и идентификаторы.

Ключевые слова – зарезервированные слова, которые имеют специальное значение для компилятора и используются только в том смысле, в котором они определены (операторы языка, типы данных и т.п.).

Идентификатор – это имя программного объекта, представляющее собой совокупность букв, цифр и символа подчеркивания. Первый символ идентификатора буква или знак подчеркивания, но не цифра. Идентификатор не может содержать пробел.

Прописные и строчные буквы в именах различаются, например, ABC, abc, Abc три различных имени. Каждое имя (идентификатор) должно быть уникальным в пределах функции и не должно совпадать с ключевыми словами.

В тексте программы можно использовать комментарии. Если текст начинается с двух символов «косая черта» // и заканчивается символом перехода на новую строку или заключен между символами /* и */, то компилятор его игнорирует.

Комментарии удобно использовать как для пояснений к программе, так и для временного исключения фрагментов программы при отладке.

Все имена, начинающиеся с двойного подчёркивания (__) или с подчёркивания и заглавной буквы, зарезервированы для реализации и не должны использоваться в качестве идентификаторов.

ЗАРЕЗЕРВИРОВАННЫЕ СЛОВА

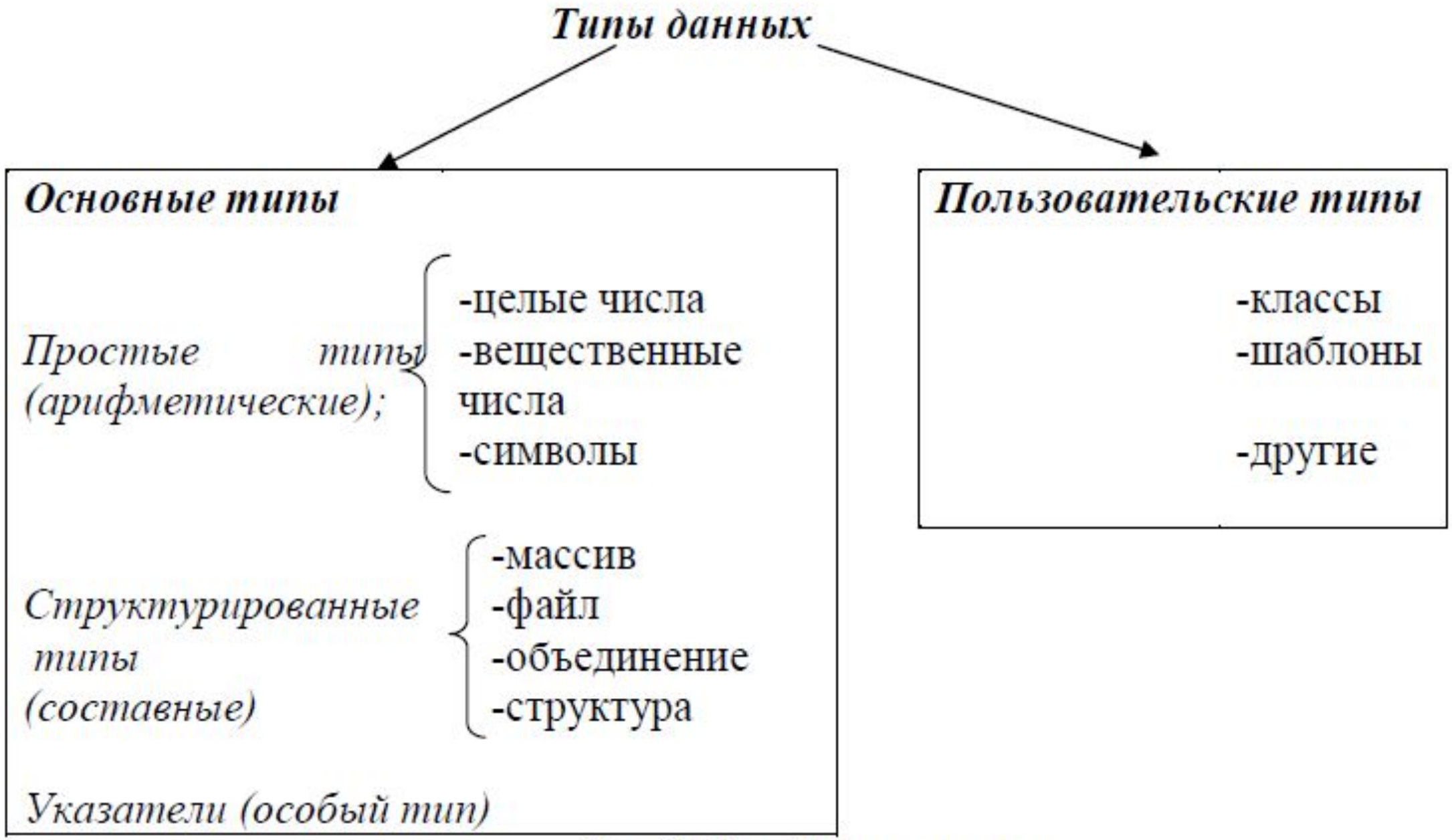
and	and_eq	asm	auto	bitand	bitor
bool	break	case	catch	char	class
compl	const	const_cast	continue	default	delete
do	double	dynamic_cast	else	enum	explicit
export	extern	false	float	for	friend
goto	if	inline	int	long	mutable
namespace	new	not	not_eq	operator	or
or_eq	private	protected	public	register	reinterpret_cast
return	short	signed	sizeof	static	static_cast
struct	switch	template	this	throw	true
try	typedef	typeid	typename	union	unsigned
using	virtual	void	volatile	wchar_t	while
xor	xor_eq				

ТИПЫ ДАННЫХ

Любая программа в процессе своего выполнения оперирует с данными. Эти данные могут быть различных типов. Типы языка C++ можно разделить на основные и составные.

К основным типам данных языка относят:

- char символьный;
- int целый;
- float с плавающей точкой;
- double двойной точности;
- bool логический.


Типы данных
Основные типы
Простые типы (арифметические);
-целые числа
-вещественные числа
-символы
Структурированные типы (составные)
-массив
-файл
-объединение
-структура
Указатели (особый тип)
Пользовательские типы
-классы
-шаблоны
-другие

Для формирования других типов данных используют основные типы и так называемые спецификаторы. Типы данных созданные на базе стандартных типов с использованием спецификаторов называют составными типами данных. В C++ определены четыре спецификатора типов данных:

- short короткий;
- long длинный;
- signed знаковый;
- unsigned беззнаковый.

Целочисленные типы:	***Размер байт/(бит)***	***Диапазон значений***
int – целый;	2/(16)	-32 768 - 32 767
short int (short) – короткий целый;	2/(16)	-32 768 - 32 767
long – длинный целый;	4/(32)	-2 147 483 648 - 2 147 483 647
unsigned – беззнаковое целое;		
unsigned int – целый б/з;	2/(16)	
unsigned short – короткий целый б/з;	2/(16)	
unsigned long – длинный целый б/з;	4/(32	0…..65 535 0…..65 535 0 4 294 967 295
Вещественные типы:		
float – одинарной точности;	4/(32)	-3.4E-38….3.4E+38
double – удвоенной точности;	8/(64)	-1.7E-308….1.7E+308
long double – максимальной точности	10/(80)	-3.4E-4932…3.4E+4932
Символьный тип		
char	1/(8)	-128….+127
unsigned char	1/(8)	0…255

Под *float* обычно отводится в два раза больше памяти чем под *int*. Под данные, описанные как *double*, отводится в два раза больше памяти чем под *float*.

Символы представляются типом *char*, под который отводится 1 байт.

К переменным целого типа относят логические переменные *bool* – *булевская переменная*, принимающая 2 значения – *true* и *false*. *enum* – *перечисляемый тип*.

Переменным, перечисленным в списке, присваиваются последовательно целые значения, начиная с нуля, заданные значения при явном присвоении.

enum { Black, Blue}; // Black =0, Blue=1

*enum qaz { ww=111, ss=222, xx=333 }; //*qaz может принимать только значения, указанные в списке.

Для корректного выполнения программы каждая переменная может использоваться для записи и хранения строго определенных типов данных. Например, если создана переменная для обработки целых чисел, то символ или вещественное число этой переменной уже присвоить нельзя.

Перед использованием переменной в программе ее необходимо заблаговременно объявить. Процедура объявления переменной предполагает, во-первых, указание типа этой переменной и, во-вторых, создание отличного от любого ключевого слова идентификатора (имени

Примеры объявления переменных:

int a; // объявлена целочисленная переменная с именем *a*

float a1, f; // объявлены две вещественные переменные *a1* и *f*.

unsigned int year=2009 ; // инициализация переменной.

На физическом уровне, объявление переменных означает, что в области оперативной памяти компьютера выделяется именованный участок памяти определенного размера, который соответствует указанному типу данных и к которому можно обратиться по имени переменной для записи или считывания данных.

Все описанные переменные в программах должны принимать какие-либо значения. Эти значения должны *присваиваться* им явно при объявлении с помощью оператора присваивания (=), например, *int a1=5*, или вычисляться в ходе выполнения программы, например, *float a2 = sqrt(a1)*.

КОНСТАНТЫ

Константы – неизменяемые величины: числа или символы, используемые в программе. Числа могут представляться в различных системах счисления.

Целые числа. Эти константы представляются типом *int*.

Десятичная константа изображается цифрами от 0 до 9. Первая цифра не может быть нулем.

Восьмеричная содержит цифры от 0 до 7, но первая цифра обязательно 0 (нуль).

Пример: $015 \rightarrow 13_{(10)}$,

$052 \rightarrow 42_{(10)}$.

Шестнадцатеричная константа начинается с комбинации символов *0x* или 0X (нуль, икс).

Пример: $0x10 \rightarrow 16_{(10)}$,

$0X25 \rightarrow 37_{(10)}$.

Константы с плавающей точкой представляются типом *double* и записываются в виде мантиссы и порядка.

Пример: 113.25e-2 → 113,25 $\cdot 10^{-2}$ → 1,1325(10),

3.7E25 → 3,7 $\cdot 10^{25}$.

Символьная константа состоит из одного символа (буква, цифра, специальный символ), заключенного в апострофы.

Пример: char SIM;

SIM=’A’;

char a=’5’;

Строку символов также относят к константам (строковым). Для их представления используются двойные кавычки:

Пример: “Error”, “125”, “d”.

При записи строковой константы в память, компилятор в ее конец помещает символ ‘\0’(*нуль-терминатор*), отмечающий конец строки.

Управляющие константы, в отличаие от простых констант, используются в связке с символом ‘\’ (обратный слэш).

\b	BS, забой
\f	Новая страница, перевод страницы
\n	Новая строка, перевод строки
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\’’	Двойная кавычка
\’	Апостроф
\\	Обратный слеш
\0	Нулевой символ, нулевой байт
\a	Звуковой сигнал

ОПРЕДЕЛЕНИЕ КОНСТАНТ С ПОМОЩЬЮ ДИРЕКТИВЫ ПРЕПРОЦЕССОРА #DEFINE

Константы в языке C++ можно задавать либо в явном виде (т.е. указывать непосредственно значение константы), либо использовать идентификатор, которому присваивается значение константы. Определение константы с помощью идентификатора осуществляется в заголовке программы по следующей форме: #define имя строка, где имя - идентификатор; строка - любая последовательность символов, отделяемая от имени хотя бы одним пробелом и заканчиваемая в текущей строке.

Директива *#define* выполняет простую текстовую подстановку, т.е. когда препроцессор встречает имя, он заменяет его на строку.

Примеры:

#define I 5 // ставит в соответствие имени I число 5

#define J 4

#define PI 3.1415

Необходимо обратить внимание на то, что при использовании директивы define тип константы не имеет значения (константы I, J, PI не имеют никакого конкретного типа). Определение констант с помощью директивы define наиболее предпочтительно, так как в случае изменения их значений в программе понадобится внести изменения только в одном месте.

ИМЕНА ПЕРЕМЕННЫХ

Переменная — поименованный участок памяти, в котором хранится значение определенного типа. У переменной есть имя (идентификатор) и значение. Имя служит для обращения к области памяти, в которой хранится значение. Во время выполнения программы значение переменной можно изменить. Перед использованием любая переменная должна быть *описана*:

тип список_переменных;

Например,

```
int a, bc, f;
float g, u, h12;
```

В C++ могут обрабатываться *структурированные типы данных*: массивы, строки, структуры, файлы.

По *месту объявления* переменные в языке C++ можно разделить на три класса: локальные, глобальные и формальные параметры функции.

Локальные переменные объявляются внутри функции и доступны только в ней.

Например:

```
int main()
{
float s; //В функции main определена вещественная переменная s,
s=4.5; //и ей присвоено значение 4.5.
}
int f1()
{
int s; //В функции f1 описана другая переменная s (типа int),
s=6; //ей присвоено значение 6.
}
int f2()
{
long int s; //В функции f2 определена еще одна переменная s
s=25; //(типа long int) и ей присвоено значение 25.
}
```

Глобальные переменные описываются до всех функций и доступны из любого места программы.

Например:

```
float s; //Определена глобальная переменная s (типа float).
int main()
{
s=4.5; //В функции main переменной s присваивается значение 4.5.
}
int f1()
{
s=6; //В функции f1 переменной s присваивается значение 6.
}
int f2()
{
s=25; //В функции f2 переменной s присваивается значение 25.
}
```

ОПЕРАЦИИ И ВЫРАЖЕНИЯ

Выражение задает порядок выполнения действий над данными и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций: a+b*sin(cos(x)). Операции делятся на унарные (например, -c) и бинарные (например, a+b).

Унарные операции	
++	увеличение значения на единицу
- -	уменьшение значения на единицу
~	поразрядное отрицание
!	логическое отрицание
-	арифметическое отрицание (унарный минус)
+	унарный плюс
&	взятие адреса
*	разадресация
(type)	преобразование типа

Бинарные операции	
+	сложение
-	вычитание
*	умножение
/	деление
%	остаток от деления
<<	сдвиг влево
>>	сдвиг вправо
<	меньше
>	больше

Бинарные операции	
<=	меньше или равно
>=	больше или равно
==	равно
!=	не равно
&	поразрядная конъюнкция (И)
^	поразрядное исключающее ИЛИ
|	поразрядная дизъюнкция (ИЛИ)
&&	логическое И
||	логическое ИЛИ
=	присваивание

Бинарные операции	
*=	умножение с присваиванием
/=	деление с присваиванием
+=	сложение с присваиванием
-=	вычитание с присваиванием
%=	остаток от деления с присваиванием
<<=	сдвиг влево с присваиванием
>>=	сдвиг вправо с присваиванием
&=	поразрядная конъюнкция с присваиванием
|=	поразрядная дизъюнкция с присваиванием
^=	поразрядное исключающее ИЛИ с присваиванием

Другие операции	
?:	условная (тернарная) операция
,	последовательное вычисление
sizeof	определение размера
(тип)	преобразование типа

ОПЕРАЦИИ ПРИСВАИВАНИЯ

Обычная операция присваивания имеет вид:

имя_переменной=значение;

где значение — это выражение, переменная, константа или функция.

Выполняется операция так:

Сначала вычисляется значение выражения указанного в правой части оператора, а затем его результат записывается в область памяти, имя которой указано слева.

Например, запись a=b означает, что переменной a присваивается значение b. Если a и b переменные разных типов, происходит преобразование типов: значение в правой части преобразуется к типу переменной левой части. Следует учитывать, что при этом можно потерять информацию или получить другое значение.

В C++ существует возможность присваивания нескольким переменным одного и того же значения. Такая операция называется множественным присваиванием и в общем виде может быть записана так:

имя_переменной1= имя_переменной2=...= имя_переменнойN=значение;

Запись a=b=c=3.14159/6; означает, что переменным a, b и c было присвоено одно и то же значение 3.14159/6.

Операции +=, -=, *=, /= называют составным присваиванием. В таких операциях при вычислении выражения стоящего справа используется значение переменной из левой части, например:

x+=p; //Увеличение x на p, то же что и x=x+p.

x-=p; //Уменьшения x на p, то же что и x=x-p.

x*=p; //Умножение x на p, то же что и x=x*p.

x/=p; //Деление x на p, то же что и x=x/p.

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Операции +, -, *, / относят к *арифметическим операциям*. Их назначение понятно и не требует дополнительных пояснений.

Операции инкремента ++ и *декремента* -- так же причисляют к арифметическим, так как они выполняют увеличение и уменьшение на единицу значения переменной. Эти операции имеют две формы записи *префиксную* (операция записывается перед операндом) и *постфиксную* (операция записывается после операнда). Так, например оператор `p=p+1;` можно представить в префиксной форме `++p;` и в постфиксной `p++;`. Эти формы отличаются при использовании их в выражении. Если знак декремента (инкремента) предшествует операнду, то сначала выполняется увеличение (уменьшение) значения операнда, а затем операнд участвует в выражении.

Например, x=12;

y=++x; //В переменной y будет храниться значение 13.

Если знак декремента (инкремента) следует после операнда, то сначала операнд участвует в выражении, а затем выполняется увеличение (уменьшение) значения операнда:

x=12;

y=x++; //Результат – число 12 в переменной y.

Остановимся на *операциях целочисленной арифметики*. Операция *целочисленного деления* / возвращает целую часть частного (дробная часть отбрасывается) в том случае если она применяется к целочисленным операндам, в противном случае выполняется обычное деление: 11/4=2 или 11.0/4=2.75. Операция *остаток от деления* % применяется только к целочисленным операндам: 11%4 = 3.

К *операциям битовой арифметики* относятся следующие операции: &, |, ^, ~, <<, >>. В операциях битовой арифметики действия происходят над двоичным представлением целых чисел.

Арифметическое И (&). Оба операнда переводятся в двоичную систему, затем над ними происходит логическое поразрядное умножение операндов по следующим правилам:

1&1=1,
1&0=0,
0&1=0,
0&0=0.

Например, если A=13 и B=23, то их двоичное представление соответственно A=0000000000001101 и B=0000000000010111. В результате логического умножения A&B получим 0000000000000101 или 5 в десятичной системе счисления. Таким образом, A&B=13&23=5.

$$\begin{array}{l} 0000000000001101 \\ \& \quad 0000000000010111 \\ \hline 0000000000000101 \end{array}$$

Арифметическое ИЛИ (|). Здесь также оба операнда переводятся в двоичную систему, после чего над ними происходит логическое поразрядное сложение операндов по следующим правилам:

1|1=1,

1|0=1,

0|1=1,

0|0=0.

Например, результат логического сложения чисел A=13 и B=23 будет равен A|B=31.

$$\begin{array}{r} 0000000000001101 \\ | \; 0000000000010111 \\ \hline 0000000000011111 \end{array}$$

Арифметическое исключающее ИЛИ (^). Оба операнда переводятся в двоичную систему, после чего над ними происходит логическая поразрядная операция ^ по следующим правилам:

1^1=0,

1^0=1,

0^1=1,

0^0=0.

Арифметическое отрицание (~). Эта операция выполняется над одним операндом. Применение операции ~ вызывает побитную инверсию двоичного представления числа ~13=14.

	0000000000001101
~a	1111111111110010