Основы языков C/C++

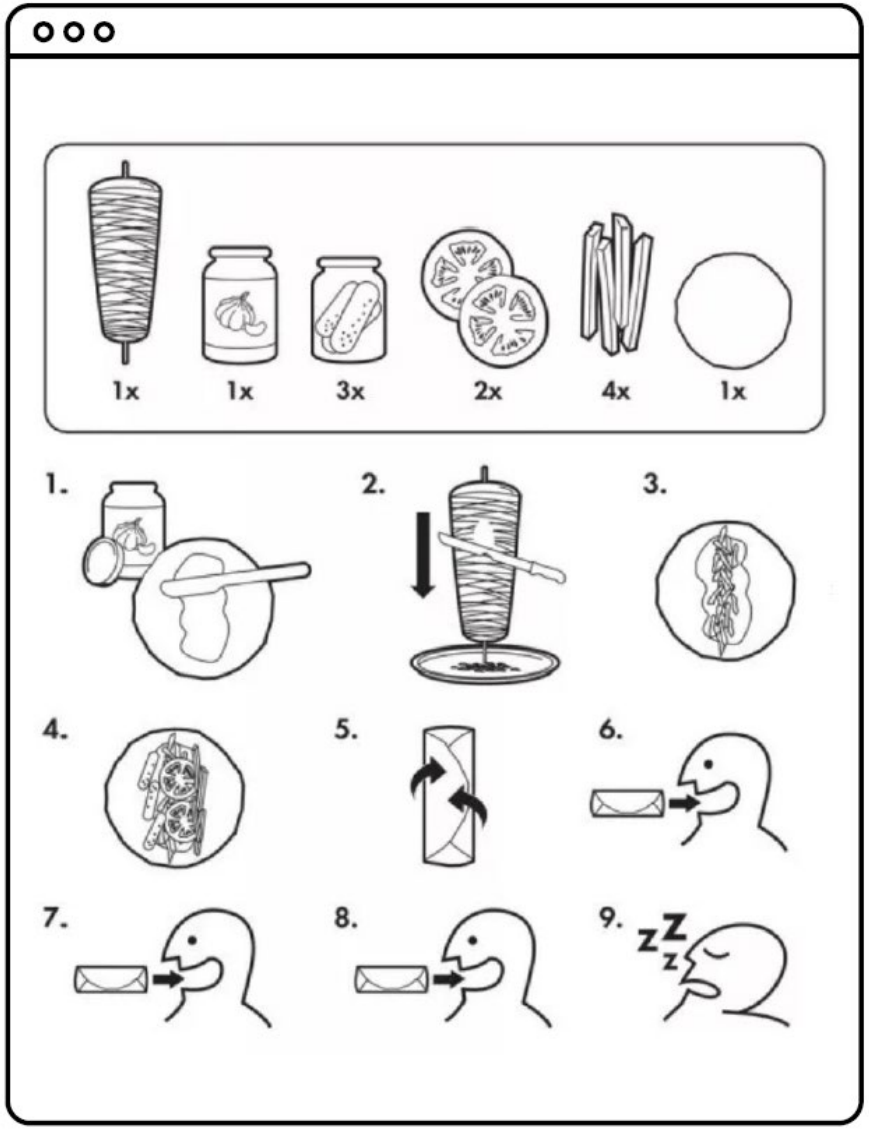
Кафедра ЭО

Петрухин О.М.

Зачем нужны программы?

Программа — это набор инструкций, выполняемый компьютером, для достижения определенной цели.

Программа — это один из способов автоматизации деятельности человека



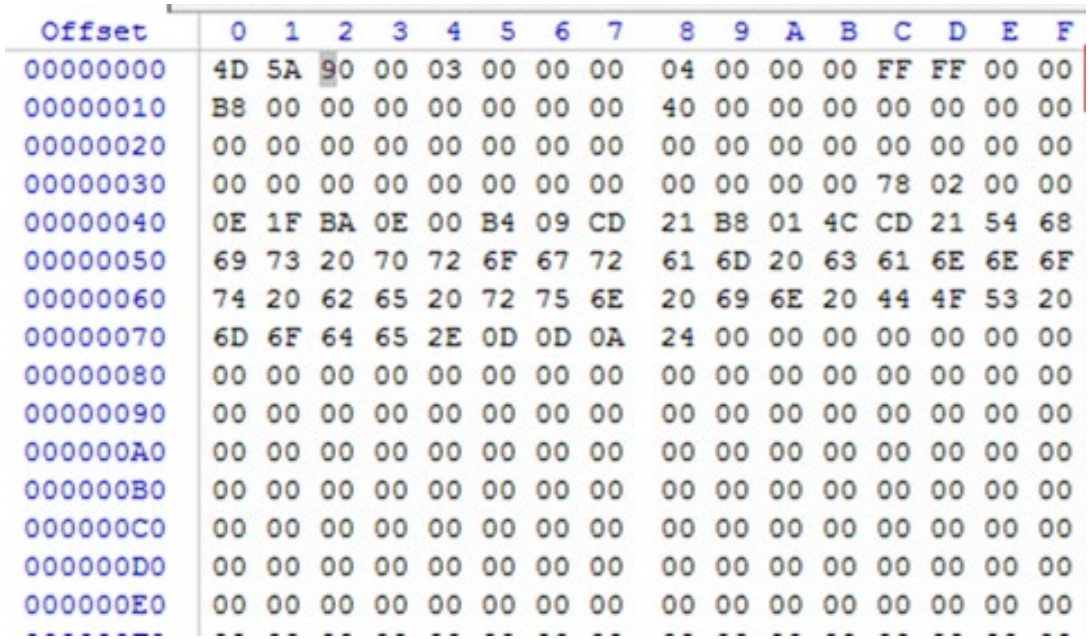
Машинный код

Это последовательность команд и их операндов, выполняемая процессором.

Процессор умеет исполнять команды, закодированные в специальном виде: машинном коде.

Это набор байт, который умеет декодировать процессор и выполнять операции:

- загрузка из ОЗУ данных в процессор
- запись данных из процессора в ОЗУ
- сложение, вычитание и т.д.
- другие операции



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	78	02	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Ассемблер

Кодировать «вручную» биты и байты достаточно сложно (мозг тяжело работает с числами), поэтому придумали специальные мнемоники, которые позволяют записывать программу в виде текста и затем «переводить» её в машинный код с помощью специальной программы — **ассемблера**

А соответствующие языки называют языками ассемблера

```
48 c7 c0 01 00 00 00    mov    $0x1,%rax
48 c7 c7 01 00 00 00    mov    $0x1,%rdi
48 c7 c6 00 20 40 00    mov    $0x402000,%rsi
48 c7 c2 0d 00 00 00    mov    $0xd,%rdx
0f 05                   syscall
48 c7 c0 3c 00 00 00    mov    $0x3c,%rax
48 c7 c7 00 00 00 00    mov    $0x0,%rdi
0f 05                   syscall
```

Языки высокого уровня

Писать на ассемблере большие программы

достаточно проблематично

Поэтому появились высокоуровневые языки,

позволяющие не думать о регистрах

процессора и системных вызовах

Пример на C++:

```
int i1, i2;
bool res;
i1 = 1 + 2 + 3;
```

Компилятор

Компиляция (compile) – процесс перевода исходного текста программы, написанной программистом, в машинный код. Или, другими словами, – создание из программы исполняемого файла

Для того, чтобы компилятор правильно и однозначно понимал то, что пишет программист, в тексте программы используются специальные выражения:

Например:

- зарезервированные слова `for`, `if`, `switch`, `int` и т.д.
- операции `+`, `-`, `*`, `/`, `<<`, `>>` ...
- операторы `sizeof`, `<<`, `>>` ...
- модификаторы `unsigned` ...

и множество других

Области применения языков C/C++

Ключевые идеи:

- **Максимальная производительность и минимальная ресурсоёмкость** при достаточно высоком уровне абстракции
- **Возможность низкоуровневого доступа** и выполнения потенциально небезопасных операций (за счёт практически полной свободы действий разработчику)
- **Прямая совместимость с языком C**: C++ сохраняет обратную совместимость с языком C, что позволяет использовать существующий C-код и библиотеки, а также интегрировать низкоуровневые решения

1. Бизнес-приложения (например, Интернет-Банкинг)
2. Сторонние библиотеки для платформы (чужой код)
3. «Платформа»: JVM/.Net/PHP/Python
4. Системные библиотеки
5. C/C++ Library & System Calls
6. Ядро ОС
7. «Железо», микропрограммы

Что нужно для написания программ на C++?

Для написания программы на любом языке (в том числе C++) требуется следующий минимальный набор инструментов:

- **текстовый редактор**

для ввода и редактирования ключевых слов, операторов и иных конструкций языка

- **компилятор**

для перевода программы с языка C++ в машинный код

- **отладчик**

для пошагового выполнения программы и выявления ошибочного поведения алгоритма

С целью сделать процесс написания, отладки и сопровождения программ максимально удобным и эффективным, существует такое понятие – **среда разработки** или **IDE** - Integrated Development Environment

- **Microsoft Visual Studio**

платный, доступен только для Windows

- **Visual Studio Code**

беспалтный, доступен для Windows, MacOS, Linux, не является полноценной средой – это редактор с элементами IDE

- **Qt Creator**

кроссплатформенный, снискал популярность из-за качественной и подробной документации к библиотекам

- **Xcode**

полноценная IDE, созданная компанией Apple для написания ПО под macOS, iOS, WatchOS и tvOS

Структура программы

```
int main() {
}
```

Это самая простая и самая маленькая программа на C++ (она не делает абсолютно ничего и поэтому совершенно бесполезна :D

```
int main()
{
// здесь нужно писать команды языка
(инструкции)
// логика инструкции всегда завершается точкой
с запятой ;
// то, что начинается с //
// и до конца строки – это комментарий

/*
ещё бывают многострочные комментарии
как сейчас
всё, что находится между символами /* и */,
также является комментарием
*/
// комментарии не являются логикой программы
// они полностью игнорируются при выполнении
// и нужны для пометок и пояснения частей кода
// если их удалить, то программа не сломается
}
```

```cpp
#include <iostream>

//using namespace std;

int main()
{
    int number1;
    int number2;
    int sum;

    //cout << "Введите первое число: ";
    std::cout << "Введите первое число: ";
    std::cin >> number1;

    std::cout << "Введите второе число: ";
    std::cin >> number2;

    sum = number1 + number2;

    std::cout << "Сумма: " << sum << std::endl;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\VsCodeProjects>  & 'c:\Users\Oleg\.vscode\ext
ensions\ms-vscode.cpptools-1.19.4-win32-x64\debugAd
apters\bin\WindowsDebugLauncher.exe' '--stdin=Micro
soft-MIEngine-In-etnjjdi4.sgLauncher.exe' '--stdin=
Microsoft-MIEngine-In-etnjjdi4.se5' '--stdout=Micro
soft-MIEngine-Out-p51x5ozg.rc5' '--stdergLauncher.e
gLauncher.exe' '--stdin=Microsoft-MIEngine-In-etnjj
di4.se5' '--stdout=Microsoft-MIEngine-Out-p51x5ozg.
rc5' '--stderr=Microsoft-MIEngine-Error-222jbxg1.vt
l' '--pid=Microsoft-MIEngine-Pid-nf3vj2lu.5uw' '--d
bgExe=C:\msys64\mingw64\bin\gdb.exe' '--interpreter
=mi'
Введите первое число: 54
Введите второе число: 46
Сумма: 100
PS C:\VsCodeProjects>
```

Переменные

Переменная — это контейнер, в котором будет находиться какое-нибудь значение.

C++ — это строго типизированный язык. Это значит, что у любой переменной есть тип и этот тип нельзя изменить. Создали переменную определенного типа — переменная будет содержать значения только этого типа до конца своей жизни.

Напоминание: не забудьте в конце поставить точку с запятой. Объявление переменной — это тоже команда.

```
int main()
{
// тип имя_переменной_1,
// имя_переменной_2, …;
int a, b, c;
char CharName;
}
```

Основные типы данных

Тип данных	Пояснение
bool	Описывает логические значения
char	Описывает символы
int	Описывает целые числа
unsigned int	Описывает целые положительные числа
float	Описывает дробные числа
void	–

`int` – от 0 до 2 147 483 647 (2^31 – 1) и от -1 до -2 147 483 648 (-2^31)

`unsigned int` – от 0 до 4 294 967 295 (2^32 – 1)

`bool` – true / false

Имена переменных

- имя может содержать цифры, английские буквы и символ подчеркивания
- первый символ должен быть буквой или символом подчеркивания
- имя не может быть ключевым словом языка C++
- имя переменной должно отображать то, что в ней хранится
- максимальный размер имени - 255 символов

Имена переменных

Язык C++ — регистрозависимый:

```
char name;
char Name;       // две переменные с разными именами
```

Сообщество C++-разработчиков придерживается нескольких стилей именования переменных:

- **CamelCase** (несколько слов пишутся слитно без пробелов, при этом каждое слово внутри фразы пишется с прописной буквы);
- **Венгерская нотация** (имена предваряются заранее оговоренными префиксами, состоящими из одного или нескольких символов)

Инициализация

```
int main()
{
    int a; // объявление неинициализированной переменной целого типа
    // если переменная не инициализирована, то в ней будет находиться какое-то
    // случайное значение
    a = 8; // в переменной a теперь находится число 8
    int a = 10; // ошибка!
    // нельзя сделать ещё одну переменную с таким же именем
    a = 10; // вот так можно. Меняем значение в существующей переменной
    double pi = 3.14; // объявление и заполнение переменной
    // программисты такое называют «инициализацией»
}
```

Ввод числовых данных с консоли

```
#include <iostream>
int main()
{
    int num;
    std::cout << "Hello!" << std::endl;
    std::cout << "Enter number: ";
    std::cin >> num;
    std::cout << "Number is: " << num << std::endl;
}
```

В данном примере мы сначала используем оператор `std::cout <<` для вывода сообщения пользователю, затем с помощью аналогичного оператора `std::cin >>` получаем пользовательский ввод.

Арифметические и логические операции C++

Алгебраическая операция	Операция на C++	Выражение C++
Сложение	+	f + 7
Вычитание	–	a – f
Умножение	*	b * m
Деление	/	x / y
Взятие по модулю	%	r % s
Операция проверки на равенство и отношение	**Операция на C++**	**Выражение C++**
Равенство	==	x == y
	!=	x != y
Отношение	>	x > y
	<	x < y
	>=	x >= y
	<=	x <=y

Арифметические и логические операции C++

```cpp
#include <iostream>
int main()
{
    std::cout << 1 + 6 << std::endl;
    std::cout << 4 - 9 << std::endl;
    std::cout << -3.5 * 3 << std::endl;
    std::cout << 7 / 5 << std::endl;
    std::cout << 7.0 / 5.0 << std::endl;
    std::cout << 5 % 3 << std::endl;
    std::cout << 2 + 2 * 2 << std::endl;
    std::cout << 1 + 6 << std::endl;
}
```

Операции присваивания, инкремента и декремента

Операция	Пример	Пояснение
Сложение	f += 7	f = f + 7
Вычитание	a -= 4	a = a – 4
Умножение	b *= 5	b = b * 5
Деление	x /= y	x = x / y
Взятие по модулю	r %= s	r = r % s
Преинкремент	++a	Значение ***a*** увеличивается на 1, затем новое значение используется в выражении, в которое входит ***a***
Постинкремент	a++	В выражении используется текущее значение ***a***, затем значение ***a*** увеличивается на 1
Предекремент	--b	Значение ***b*** уменьшается на 1, затем новое значение используется в выражении, в которое входит ***b***
Постдеремент	b--	В выражении используется текущее значение ***b***, затем значение ***b*** уменьшается на 1

Операции присваивания, инкремента и декремента

```cpp
#include <iostream>
int main()
{
    int num = 10;
    std::cout << num++ << std::endl;
    std::cout << ++num << std::endl;
    std::cout << num-- << std::endl;
    std::cout << --num << std::endl;
    std::cout << (num += 5) << std::endl;
    std::cout << (num -= 6) << std::endl;
    std::cout << (num *= 9) << std::endl;
    std::cout << (num /= 10) << std::endl;
    std::cout << (num %= 3) << std::endl;
}
```

```cpp
#include <iostream>
int main()
{
    int16_t c = 5;
    std::cout << c << std::endl;  // Печатает 5
    // Печатает 5, затем выполняет инкремент
    std::cout << c++ << std::endl;
    std::cout << c << std::endl;  // Печатает 6
    // Выполняет инкремент, затем печатает 7
    std::cout << ++c << std::endl;
    std::cout << c << std::endl;  // Печатает 7
}
```

Приоритеты операций

Операция	Тип
()	Круглые скобки
++ --	Унарные (постфиксные)
++ --	Унарные (префиксные)
* / %	Мультипликативные
+ -	Аддитивные
<< >>	Передачи/извлечения
< <= >= >	Отношения
== !=	Равенства
?:	Условная
= += -= *= /= %=	Присваивания

Приведение типа

Рассмотрим такой пример:

```
char num = 4;
char num1 = '4';
std::cout << num + num1 << std::endl;
```

Вопрос: Что будет выведено на экран?

Ответ: 56

Приведение типа (typecast) — преобразование значения одного типа в значение другого типа.

Преобразование типа может быть:

- **явным** (указывается программистом как отдельная команда);
- **не явным** (выполняется самим компилятором).

Явное приведение типа

- круглые скобки
- static_cast
- dynamic_cast
- const_cast
- reinterpret_cast

Мы рассмотрим только первые два

```
(int)num // старый вариант, оставлен для совместимости с языком C
static_cast<int>(num) // современный вариант, входит в стандарт C++
```

static_cast < тип >(объект):

- тип — тип переменной который нужно получить;
- объект — переменная, тип которой нужно преобразовать.

Явное приведение типа

```
#include <iostream>
int main()
{
    char num = 4, num1 = '4';
    double fl = 5.6, fl1 = 5.7;
    std::cout << (num + num1) << std::endl; // 56
    std::cout << num << std::endl; //
    std::cout << static_cast< int >(num) << std::endl; // 4
    std::cout << fl << std::endl; // 5.6
    std::cout << fl1 << std::endl; // 5.7
    std::cout << fl + fl1 << std::endl; // 11.3
    std::cout << static_cast<int>(fl + fl1) << std::endl; // 11
    std::cout << static_cast<int>(fl)+static_cast<int>(fl1)<< std::endl; // 10
    std::cout << static_cast<int>(num + num1) << std::endl; // ?
}
```

КАИ
Крылья для жизни!

Спасибо за внимание!