

# Язык программирования BCPL

Канарейкин А. И.

# История

BCPL (аббр. Basic/Bootstrap CPL) — бестиповый структурный язык программирования, разработанный Мартином Ричардсом (Martin Richards) в 1966 году (первая реализация создана им же в 1967).



Язык был создан на основе CPL путем удаления из него элементов, усложняющих компиляцию, а именно: типизации переменных и набора синтаксических структур, определяющих блоки с различными правилами видимости переменных. Кроме того, синтаксис языка значительно упрощен и добавлена возможность раздельной компиляции сегментов программы.

BCPL разрабатывался как компактный язык для системного программирования и написания компиляторов, в частности, на нем должен был быть написан компилятор CPL (отсюда вариант названия Bootstrap CPL). В итоге первый компилятор, переписанный на BCPL же, занимал около 1000 строк и требовал менее 20 кБ памяти для запуска.

В свое время язык использовался достаточно широко: к 1979 году существовали компиляторы для как минимум 25 архитектур, на нем даже писали операционные системы (TRIPOS и AmigaDOS). Он оказал значительное влияние на B, который, в свою очередь, стал прародителем C. Именно из BCPL создатели этих языков позаимствовали идею предоставления программисту свободы, не ограниченной возможностями языка. В настоящее время язык не используется в практических целях; существует только одна реализация языка, созданная Мартином Ричардсом и предназначенная для Cintpos (варианта TRIPOS).



Для BCPL был впервые придуман и реализован ряд возможностей, ставших характерными для более современных языков. В частности, именно в нем были впервые использованы фигурные скобки { } для выделения блоков и // для выделения комментариев до конца строки. Считается, что именно на BCPL была написана первая программа “Hello, World!”. Для обеспечения переносимости языка и программ, написанных на нем, между разными архитектурами, была изобретена виртуальная машина. Компилятор состоял из двух частей: фронт-энда, который разбирал код программы и генерировал по нему O-CODE (объектный код), и бэк-энда, который интерпретировал объектный код или транслировал его в машинно-зависимый. Для переноса языка на новую архитектуру было достаточно переписать бэк-энд, а он занимал около 1/5 общего объема компилятора.

Единственный тип данных в языке — целое число, обычно 16-битное, позднее 32-битное. При этом применяется контекстная типизация — переменные трактуются как числа, символы, указатели, битовые паттерны или логические значения в зависимости от того, какие операторы к ним применяются. Проверки того, может ли конкретная операция применяться к конкретному значению переменной, не проводятся — за этим должен следить программист. Некоторые реализации добавляли в язык числа с плавающей точкой и специальные операторы для их обработки.

Единственная структура данных языка — массив (последовательность ячеек). Используя ячейки массива в качестве указателей на другие переменные/массивы, можно строить более сложные структуры.

Структура блоков и управляющих команд BCPL близка к используемой в CPL. Блок может использоваться вместо команды или вместо выражения; во втором случае перед блоком добавляется слово VALOF, а результат его выполнения задается словом RESULTIS внутри блока. В любом блоке могут присутствовать объявления локальных переменных и процедур. Позднее язык был расширен возможностью определения сопрограмм, что сделало системное программирование еще более удобным.

Имена процедур, если рассматривать их как значения, являются указателями на соответствующие фрагменты кода. Они могут храниться в переменных, передаваться в функции/процедуры как аргументы и т.д. Возможны переходы между процедурами при помощи псевдофункций level() и longjmp(). Принципы работы с процедурами позднее были заимствованы для C.

Набор библиотек языка предельно лаконичен: они содержат только базовые функции ввода-вывода данных и динамического распределения памяти.

# Элементы синтаксиса:

Комментарий до конца строки	//
Регистрозависимость	да
Присваивание значения переменной	<varname> := <value>, <varname> : <value>
Объявление переменной с присваиванием значения	let varname = value
Группировка выражений	( ... )
Блок	{ ... } или \$( ... \$)
Равенство	=
Неравенство	~= neqv
Определение функции	f(para1, para2, ...) = valof \$( ... \$) или let f(para1, para2, ...) be \$( ... \$)
Вызов функции	f(a,b,...f) или f[a,b,...]
Если - то	if <condition> do <trueBlock>
Если - то - иначе	test <condition> then <trueBlock> or <falseBlock>; <condition> -> <trueBlock>, <falseBlock>
Цикл с предусловием	while <condition> do <loopBody>
Цикл с постусловием	<loopBody> repeatuntil <condition>
Цикл for - next для диапазона целых чисел с инкрементом на 1	for i = 1 to 10 do <loopBody>