DOI 10.37882/2223–2966.2021.02.30

ИСПОЛЬЗОВАНИЕ ПРЕИМУЩЕСТВ НИЗКОУРОВНЕВЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ ДЛЯ НОВОГО РАЗРАБАТЫВАЕМОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ

TAKING ADVANTAGE OF LOW-LEVEL PROGRAMMING LANGUAGES FOR A NEW PROGRAMMING LANGUAGE

A. Tretyak
I. Kapetsky
E. Vereshchagina

Summary. The article discusses the concept of low-level programming languages and their advantages relative to high-level languages. When developing programs in low-level languages the developer gains access to all the capabilities of the processor, so using low-level languages efficient and compact programs can be created, what are the main advantages of low-level languages. To use these advantages in the developed programming language, it is proposed to translate its code into C ++ with further compilation of the C ++ code by a C++ compiler. As an example, this article compares the performance of the high-level programming language Python with a new programming language, which is also high-level, but has the benefits of low-level languages.

Keywords: low-level programming language, high-level programming language, programming language development, compiler, machine code.

Третьяк Александр Викторович
Аспирант, Дальневосточный федеральный университет, г. Владивосток
alextretyak2@gmail.com
Капецкий Игорь Олегович
Старший преподаватель, Дальневосточный федеральный университет, г. Владивосток
ikapeckij@mail.ru
Верещагина Елена Александровна
К.т.н., доцент, Дальневосточный федеральный университет, г. Владивосток
everesh@mail.ru

Аннотация. В статье рассматривается понятие низкоуровневых языков программирования и их преимущества относительно высокоуровневых языков. Поскольку при разработке программ на языках низкого уровня разработчик получает доступ ко всем возможностям процессора, с помощью языков низкого уровня создаются эффективные и компактные программы, что является основными преимуществами низкоуровневых языков. Для использования данных преимуществ в разрабатываемом языке программирования предлагается производить его трансляцию в C++ с дальнейшей компиляцией кода C++ компилятором. В качестве примера приводится сравнение производительности высокоуровневого языка Python с новым разрабатываемым языком программирования, который также является высокоуровневым, но обладает преимуществами низкоуровневых языков.

Ключевые слова: низкоуровневый язык программирования, высокоуровневый язык программирования, разработка языка программирования, компилятор, машинный код.

Введение

Язык программирования — это формальный язык, предназначенный для записи компьютерных программ [1]. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит компьютер под её управлением. Язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами и т.п. [2, с. 7].

В настоящее время существуют тысячи языков программирования. Их можно классифицировать различными способами. Один из способов классификации — по аппаратноориентированности и человекоориентированности — низкоуровневые и высокоуровневые языки программирования [3].

Низкоуровневый язык программирования (язык программирования низкого уровня) — язык для написания компьютерных программ, похожий на машинный код [4].

Высокоуровневый язык программирования (язык программирования высокого уровня) — язык для напи-

сания компьютерных программ, который больше похож на человеческий, чем на компьютерный язык, за счет чего его легче понять программисту [5].

Целью данной работы является исследование преимуществ низкоуровневых языков программирования и использование этих преимуществ в новом разрабатываемом языке программирования.

Преимущества низкоуровневых языков программирования

С помощью языков низкого уровня создаются эффективные и компактные программы, поскольку разработчик получает доступ ко всем возможностям процессора. Под эффективной понимается программа, имеющая высокую скорость выполнения и малый объём занимаемой оперативной памяти, а под компактной — программа, имеющая небольшой размер скомпилированного исполняемого файла.

Среди недостатков низкоуровневых языков: необходимость высокой квалификации программиста ввиду сложности программ, труднопереносимость программ на устройство с другим типом процессора, значительное время разработки больших программ [6]. Всех этих недостатков лишены высокоуровневые языки программирования.

Таким образом, основными преимуществами низкоуровневых языков являются скорость выполнения и компактность программ.

Низкоуровневые характеристики языка C

Конструкции языка C близко сопоставляются типичным машинным инструкциям, благодаря чему он нашёл применение в проектах, для которых был свойственен язык ассемблера — в операционных системах, системном и прикладном программном обеспечении, а его синтаксис стал основой для таких языков программирования, как C++, C#, Java и Objective-C [7].

C-подобные языки C и C++ наиболее близки к низкоуровневым языкам из всех прочих языков высокого уровня и имеют следующие характеристики.

Во-первых, язык C обладает портируемостью. Портируемость компилятора упрощает разработку компиляторов языка для новых платформ, а многообразие поддерживаемых компиляторами платформ избавляет разработчиков от необходимости переписывать прикладные программы для каждого компьютера.

Во-вторых, язык C позволяет работать с деталями реализации платформы: непосредственная работа с памятью и объектами в ней как с массивом байтов, возможность напрямую работать с адресами байтов и развитая арифметика указателей [8].

В-третьих, язык C достаточно близок к аппаратной части, поэтому он разделяет такие преимущества низкоуровневых языков, как скорость выполнения и компактность программ.

Вторая и третья представленные характеристики полностью соответствуют исключительно низкоуровневым языкам программирования.

Пример ускорения программного кода

Для использования преимуществ низкоуровневых языков программирования, таких как скорость выполнения и компактность программы, в разрабатываемом новом языке производится его трансляция в C++ с дальнейшей компиляцией кода C++ компилятором.

Однако во многом синтаксис языка Python удобнее для восприятия программистом, чем языков C и C++ (разработка на Python идёт быстрее, чем на многих других языках), а также чаще используется при решении таких задач как написание скриптов для обработки файлов и веб-запросов, для автоматизации задач в системном администрировании, а также для анализа больших данных. Также Python подходит и для создания прикладных приложений или игр [9]. Поэтому в разрабатываемом новом языке программирования синтаксис основывается на языке Python, но при этом сохраняются преимущества низкоуровневых языков, такие как скорость выполнения и малый объём занимаемой памяти.

Ниже представлен пример исходного кода утилиты, проверяющей парность скобок и кавычек во всех текстовых файлах в заданном каталоге.

Код на Python:

```
import sys, os

if __name__ == '__main__':
for root, dirs, files in os.walk(sys.argv[1]):
for name in files:
if (name.endswith('.txt') and name not in ('last-message.txt', 'cur-message.txt', 'histedit-last-edit.txt')) or name.endswith(('.py', '.hpp', '.h', '.cpp')):
if b"\r" in open(os.path.join(root, name), 'rb').read():
sys.exit(R"\r found in file '" + os.path.join(root, name) + "'")
```

```
    filestr: str
    try:
    filestr = open(os.path.join(root, name), 'r', encoding = 'utf-8-sig').read()
    except:
    sys.exit("Exception while reading file '" + os.path.join(root, name) + "'")
    for pair in ('"', '()', '{}', '[]'):
    i = 0
    while i < len(filestr):
    if filestr[i] == pair[0]:
    start_i = i
    nesting_level = 1
    i += 1
    while True:
    if i == len(filestr):
    sys.exit("Balance check error in file '" + os.path.join(root, name) + "'")
    ch = filestr[i]
    i += 1
    if ch == pair[0]:
    nesting_level += 1
    elif ch == pair[1]:
    if pair[0] == '(':
    assert(pair[1] == ')')
    if filestr[i-1: i] == ':' and filestr[i+1: i+3] == '(:':
    assert(filestr[i] == ')')
    i += 2 # пропускаем, чтобы смайлы:)(: не [ломали/] портили баланс
    continue
    nesting_level -= 1
    if nesting_level == 0:
    break
    elif filestr[i] == pair[1]:
    if pair[0] == '(':
    assert(pair[1] == ')')
    if filestr[i-1: i] == ':' and filestr[i+1: i+3] == '(:':
    assert(filestr[i] == ')')
    i += 2 # пропускаем, чтобы смайлы:)(: не [ломали/] портили баланс
    continue
    sys.exit("Balance check error in file '" + os.path.join(root, name) + "'")
    else:
    i += 1
```

Код на новом языке программирования:

```
    : start:
    loop(_fname) fs: walk_dir(: argv[1], files_only' 0B)
    var root = fs: path: dir_name(_fname)
    [String] dirs, files
    if fs: is_dir(_fname) {dirs [+]= fs: path: base_name(_fname)} else files [+]= fs: path: base_name(_fname)
    loop(name) files
    if (name.ends_with('.txt') & name!in ('last-message.txt', 'cur-message.txt', 'histedit-last-edit.txt')) | name.ends_with(('.py', '.hpp', '.h', '.cpp'))
    if "\r".code in File(fs: path: join(root, name)).read_bytes()
    exit('\r found in file ''fs: path: join(root, name)"')
    String filestr
    exception.try
    filestr = File(fs: path: join(root, name), 'r', encoding''utf-8-sig').read()
    exception.catch
    exit('Exception while reading file ''fs: path: join(root, name)"')

    loop(pair) ('"', '()', '{}', '[]')
    var i = 0
    loop i < filestr.len
    if filestr[i] == pair[0]
    var start_i = i
    var nesting_level = 1
    i++
    loop
    if i == filestr.len
    exit('Balance check error in file ''fs: path: join(root, name)"')
    var ch = filestr[i]
    i++
    if ch == pair[0]
    nesting_level++
    else if ch == pair[1]
    if pair[0] == '('
    assert(pair[1] == ')')
    if filestr[i — 1 .< i] == ':' & filestr[i + 1 .< i + 3] == '(:'
    assert(filestr[i] == ')')
    i += 2
    loop.continue
    nesting_level—
    if nesting_level == 0
    loop.break
    else if filestr[i] == pair[1]
    if pair[0] == '('
    assert(pair[1] == ')')
    if filestr[i — 1 .< i] == ':' & filestr[i + 1 .< i + 3] == '(:'
    assert(filestr[i] == ')')
    i += 2
    loop.continue
    exit('Balance check error in file ''fs: path: join(root, name)"')
    else
    i++
```

Данная утилита запускалась в каталоге с исходным программным кодом транспайлера нового языка программирования [10]. Время работы данной утилиты на Python составило 4,5 секунды, а скомпилированного кода на новом языке — 0,26 секунд. Таким образом, ско-

рость работы данной утилиты в 17 раз выше на новом языке программирования по сравнению с языком Python.

Заключение

Основными преимуществами низкоуровневых языков программирования являются скорость выполнения и компактность программы; эти языки ориентированы на высокую производительность, эффективное использование аппаратных ресурсов компьютера, такие как процессор, память.

Для использования данных преимуществ в разрабатываемом языке программирования предлагается производить его трансляцию в C++ с дальнейшей компиляцией кода C++ компилятором.

Таким образом, можно сделать вывод, что используя трансляцию нового разрабатываемого языка программирования в C++, увеличивается производительность выполнения программного кода, а также используется малый объём занимаемой оперативной памяти.

ЛИТЕРАТУРА

1. Systems and software engineering — Vocabulary. URL: https://www.iso.org/standard/50518.html
2. Молдованова, О. В. Языки программирования и методы трансляции: учебное пособие / О. В. Молдованова. — Новосибирск: Сибирский государственный университет телекоммуникаций и информатики, 2012. — 134 с. — URL: http://www.iprbookshop.ru/54809.html
3. Белова И. Язык программирования Python URL: https://youtu.be/6I7ybevPUKM?t=42
4. Low-level language URL: https://dictionary.cambridge.org/us/dictionary /english/low-level-language
5. High-level language URL: https://dictionary.cambridge.org/us/dictionary/ english/high-level-language
6. Языки программирования низкого уровня URL: https://studfile.net/preview/3972576/page:3/
7. Си (язык программирования) URL: https://ru.wikipedia.org/ wiki/Си_(язык_программирования)
8. И всё же C — низкоуровневый язык URL: https://habr.com/ru/company/badoo/blog/465429/
9. Где используется Python и в чём особенности этого языка URL: https://geekbrains.ru/posts/dlya-chego-nuzhen-yazyk-python
10. GitHub: 11l_to_cpp URL: https://github.com/11l-lang/_11l_to_cpp

Дальневосточный федеральный университет