DOI 10.34185/1562-9945-5-148-2023-02
УДК 004.052

В.В. Герасимов, Н.В. Карпенко, І.А. Скуратовський

ДОСТУП ДО ЕЛЕМЕНТІВ СТРУКТУРИ І НЕВИЗНАЧЕНА ПОВЕДІНКА КОДУ НА МОВІ С

Анотація. Досліджено доступ до змінних типу даних структура на мові програмування С. Проаналізовано особливості невизначеної поведінки коду після компілятору Visual Studio 2019 під керуванням Windows 10 та компілятору gcc під керуванням ОС Linux. Показано результати нехтування повідомленнями при компіляції програм, при яких останні пра цюють некоректно. Надано рекомендації щодо уникнення цієї проблеми.
Ключові слова: невизначена поведінка, структура, компілятор, мова програмування С.

Постановка проблеми. Під час розробки програмного забезпечення розробники-початківці зазвичай отримують багато повідомлень про помилки та просто попереджень різного виду. І якщо є помилки, то код просто не запуститься, але при наявності різноманітних попереджень програма зазвичай запускається. І тут важливо розуміти, до яких наслідків може привести наявність попереджень різного роду.

Аналіз останніх досліджень і публікацій. Дуже часто попередження компілятора дратують програмістів і вони шукають в Інтернеті рішення для усунення цих попереджень через налаштування компілятора. Але досвідчені програмісти скажуть, що попередження ігнорувати не можна ні в якому випадку, оскільки вони можуть бути настільки важливими, що код написаний таким чином буде умовно робочим.

Інформація про те, які попередження існують в IDE Visual Studio, та які з них вимкнені за замовченням, можна знайти на сайті Microsoft [1]. Стаття [2] надає рекомендації щодо налаштування рівня попереджень для середовищ розробки Visual Studio, Code::Blocks та GCC/G++. Також тема про попередження компіляторів піднімається в книзі про покращення коду, написаного на мові C++ автора Scott Meyers [3]. Досить цікавою і корисною є стаття з прикладами усунення помилок сегментації для мов програмування C/C++ [4]. Однак основна маса статей містить лише загальні відомості без прикладів. І коли програ-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

міст стикається з чимось незрозумілим, він починає шукати відповідь і не знаходячи її, залишає своє питання в професійних блогах чи на форумах, але на деякі питання так і немає відповідей.

Метою даної роботи є дослідження невизначеної поведінки коду при роботі зі структурами на мові програмування С при видачі відповідного попередження компілятора про повернення адреси локальної або тимчасової змінної.

Основна частина. Студенти, які тільки почали програмувати, фокусуються на тому, щоб написана програма працювала і перевіряють її на обмеженій кількості комплектів тестових даних. Це приводить до того, що вони можуть навіть і не помітити, коли в програмі є невизначеність поведінки [5].

В процедурній мові програмування С є пращур класу ООП — структура struct, яка інкапсулює лише стан сутності. І виникає питання — а чи можна працювати з окремими компонентами-полями такої структури аналогічно мовам ООП?

Розглянемо проблему доступу до елементів структури на мові С на прикладі найпростішої інформаційної системи "Телефонний довідник", де в якості структури даних будемо використовувати наступну структуру [6] (лістинг 1).

Лістинг 1

```
typedef struct {
 int id;
 char firstName[20];
 char lastName[20];
 char number[20];
} person;
```

Для оптимальної реалізації функціоналу пошуку за ім’ям, прізвищем або телефоном потрібно загальну частину оформити у вигляді окремої функції, в якій певна частина структури буде порівнюватись із введеним текстом. В об’єктно-орієнтованих мовах програмування високого рівня (C#, Java тощо) для отримання частини структури (класу) зазвичай використовують так звані гетери — спеціальні функції, призначені для повертання значень полів екземпляру класу (його стану). Спробуємо реалізувати щось подібне на мові С (лістинг 2).

Лістинг 2

```
char* getFirstName(person person)
{
 return person.firstName;
}
```

Ця функція повертає ім’я людини. Аналогічним чином можна реалізувати функції для виокремлення прізвища та номера телефону. Тоді загальна функція для пошуку за будь-якою частиною структури буде мати наступний вигляд (лістинг 3).

Лістинг 3

```
    void searchByPartInfo(char* part, char* (*func)(person per-
son))
    {
     char arr[FIELD], * info;
     printf("Enter %s\n", part);
     scanf_s("%s", arr, FIELD);
     int counter = 0;

     for (int i = 0; i < N; i++)
     {
       if (massiveStruct[i].id != -1)
       {
         if (strstr(func(massiveStruct[i]), arr))
         {
           printItem(i);
           counter++;
         }
       }
     }

     if (counter == 0)
     {
       puts("Not found");
     }
    }
```

де char* (*func)(person person) — вказівник на наш псевдогетер, який повертатиме необхідну частину структури даних.

При компіляції вихідного коду в середовищі Visual Studio видаються попередження відносно наших псевдогетерів — warning C4172: *повернення адреси локальної або тимчасової змінної: person*. Але тим не менш, код запускається і працює, що може приспати пильність розробника коду, особливо недосвідченого. А ось розробники зі стажем можуть відразу згадати — Undefined behaviour, невизначена поведінка. В чому ж проявляється ця невизначена поведінка?

Тестування коду показало, що пошук за ім’ям і прізвищем в такому випадку працює бездоганно, але за номером телефону (який стоїть на останньому місці під час оголошення структури) — не працює зовсім. Щоб зрозуміти про-

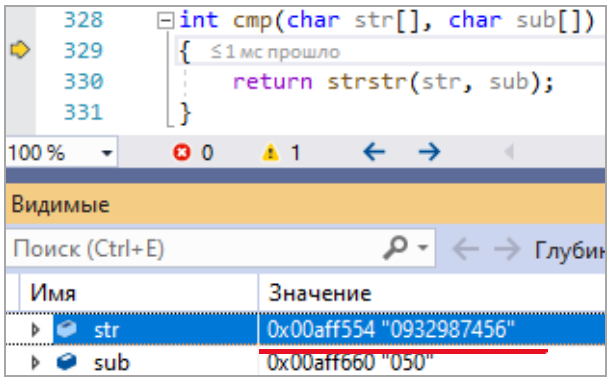
ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

блему необхідно за допомогою налагоджувача (debugger) прослідкувати за ходом виконання коду. Для цього створимо невелику функцію-обгортку (лістинг 4).

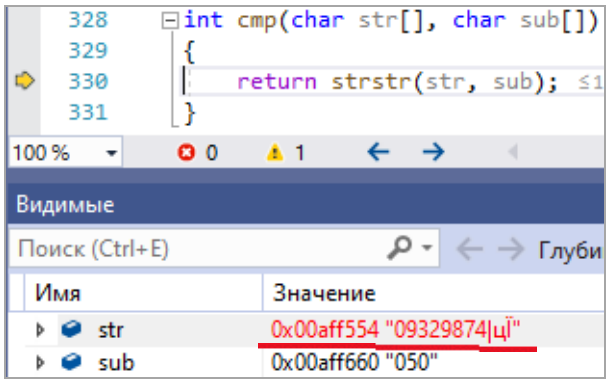
Лістинг 4

```
int compare(char str[], char sub[])
{
 return strstr(str, sub);
}
```

Відразу після додавання такої функції-обгортки картина дещо змінилась, а саме в деяких випадках пошук почав виконуватись. Детальний аналіз показав, що відразу після входу в функцію-обгортку символьний рядок, який містить номер телефону залишається без змін (рис. 1,а). Але функція для порівняння символьних рядків strstr повертає рядок, в якому без змін залишаються перші 8 цифр, інші — змінюються випадковим чином (рис. 1,б). Тому якщо область пошуку обмежена тільки першими 8 цифрами — результат успішний.



а)



б)

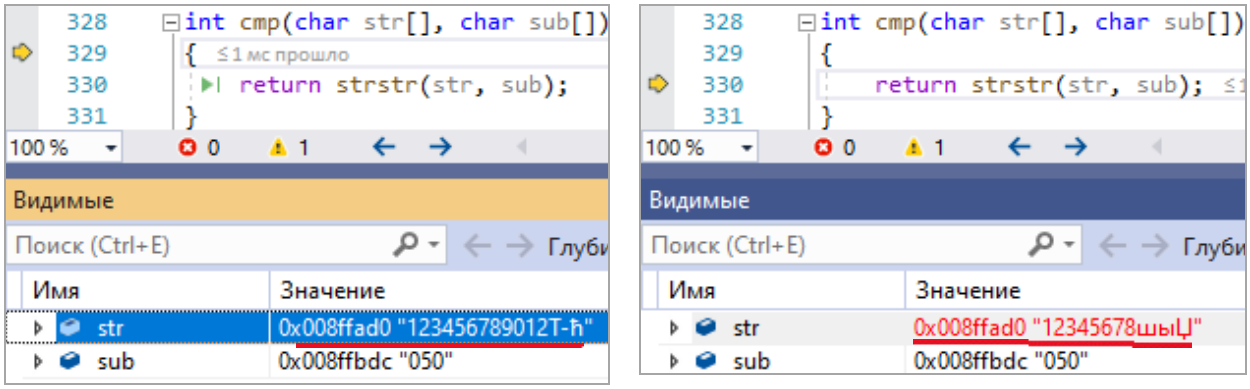
Рисунок 1 – Виявлення моменту заміни символів початкового рядку:
а) вхід в функцію-обгортку; б) результат, який повертає функція strstr

Були спробувані різні варіанти вирішення проблеми. Масив оголошувався і як статичний, і як константний — жодне рішення не допомогло вирішити проблему. Незмінним залишалось лише одне — перші 8 цифр. Решта масиву або змінювалась випадковим чином, або просто відкидалась. Передати на обробку частину структури у вигляді масиву символів-цифр не вдавалось.

Потім були проведені дослідження як змінюється значення останнього члена структури в залежності від довжини масиву (char number[20]). Так при наявності 19 цифр (останній байт відводиться під символ закінчення рядку) в цій частині структури, яку ініціалізували цифрами 1234567890123456789, був отриманий результат, представлений на рис. 2, а саме: при вході в функцію-

оболонку масив обрізається до 12 символів (рис. 2,а), а на наступному кроці — обрізається ще раз до 8 символів і доповнюється випадковими символами (рис. 2,б).

Подібні експерименти з іншими членами структури, які стоять не на останньому місці, до таких результатів не призвели. Тобто така поведінка при передачі між функціями спостерігається лише для останнього члена структури, який є символьним масивом.



а) б)

Рисунок 2 – Заповнення останнього члена структури цифрами на максимальну ємність: а) значення на вході в функцію-оболонку; б) результат, який повертає функція strstr

Також були проведені дослідження для різної кількості байт пам'яті, яку програміст відводить для останнього члена структури під час оголошення. Так, у випадку розміру масиву в 10 символів (char number[10]) при вході у функцію-оболонку залишалось лише 4 символи, а на наступному кроці початкових символів вже не залишалось зовсім, були лише випадкові символи. У випадку масиву в 40 символів (char number[40]) при вході у функцію-оболонку залишалось 32 символи, а на наступному кроці їх залишалось лише 28.

Переміщення цілого поля структури в її кінець (поля id) в цілому картину не змінило — при передачі в якості аргументу останнього символьного масиву спостерігається втрата інформації, змінюється лише кількість символів, які залишаються без змін.

Дослідження були проведені в середовищі Visual Studio 2019 під керуванням операційної системи Windows 10, коли в налаштуваннях проекту був обраний стандарт мови С за умовчанням — MSVC. Коли стандарт був змінений на більш сучасний стандарт ISO C17 (2018) [7] — в останній частині структури, яка

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

відповідала за номер телефону, незмінними залишались лише 4 цифри, інші змінювались випадковим чином.

Таким чином ми бачимо дійсно невизначену поведінку, коли результат виклику функції залежить від багатьох чинників: від довжини масиву, від стандарту мови С, від положення масиву в структурі.

А ось спроба провести аналогічні дослідження під керуванням операційної системи Linux Mint з використанням компілятору gcc версії 5.4 не вдалась. При компіляції коду також видавалось попередження про повернення адреси локальної змінної. А ось при спробі здійснити пошук за допомогою вищенаведеного коду за будь якою частиною структури даних (ім'я, прізвище чи телефон) програма просто аварійно закінчувала роботу з повідомленням про помилку сегментування [4]. Відлагодження вихідного коду показало, що у функцію-оболонку передається пустий вказівник, тобто наш "псевдогетер" не повертає взагалі нічого.

Пояснення такої поведінки є досить простим, для цього потрібно лише детально розглянути наш псевдогетер (лістинг 2). В мові програмування С структура передається в метод за значенням, а це означає, що при вході в функцію ця структура буде скопійована в локальну змінну, отже всі поля цієї структури будуть скопійовані в локальній області видимості (в межах методу). При цьому в самій структурі в нас фігурують не вказівники, а саме масиви символів, і тому наші поля структури будуть скопійовані за новими адресами. В псевдогетері ми повертаємо вказівник на перший елемент масиву символів, який після виходу за межі області видимості функції буде помічений як видалений. Тобто система після виходу з методу вважає, що може знову використовувати область пам’яті, яку займав цей масив символів. Отже на наступних ітераціях в цю частину пам’яті можуть бути записані інші дані, причому не обов’язково тієї самої довжини, і збереження значення змінної за цією адресою не гарантується. Ми можемо спостерігати як правильну поведінку, так і повний або частковий перезапис даних.

Всі ці дослідження були проведені для того, щоб з’ясувати — в чому саме полягає невизначена поведінка коду? Ну а для того, щоб все-таки вирішити поставлену задачу доступу до елементів структури на мові С в псевдогетерах та інших подібних випадках необхідно передавати структуру за адресою (посиланням). Або як у випадку найпростішої інформаційної системи можна скористатись глобальним масивом структур і оперувати відповідно тільки з індексами цього масиву [6].

Висновки. У роботі проведено дослідження невизначеної поведінки коду, написаного на мові програмування С, при передачі членів структури в функцію.

Компілятори Visual Studio 2019 і gcc надавали повідомлення про невизначену поведінку коду. Але ця поведінка кардинально відрізнялась для різних операційних систем та різних компіляторів. Якщо після gcc під ОС Linux код просто зовсім не працює і програма зупиняє свою роботу з повідомленням про помилку сегментації, то після Visual Studio під керуванням Windows код запускався, але працював коректно лише на обмежених комплектах даних.

Було показано «підводні камені» різноманітних рішень для виправлення цієї помилки. Таким чином, невизначена поведінка коду усувається передачею в функцію адреси структури, з членом якої потрібно працювати.

ЛІТЕРАТУРА

1. Ошибки и предупреждения компилятора и средств сборки C/C++. URL: https://learn.microsoft.com/ru-ru/cpp/error-messages/compiler-errors-1/c-cpp-build-errors?view=msvc-170

2. Конфигурация компилятора: Уровни предупреждений и ошибки. URL: https://ravesli.com/konfiguratsiya-kompilyatora-urovni-preduprezhdenij-i-oshibki/

3. Мэйерс С. Эффективное использование C++. 55 верных способов улучшить структуру и код ваших программ. – М.: ДМК Пресс, 2006. – 300 с. – ISBN 5-94074-304-8, 0-321-33487-6.

4. Segmentation Fault in C/C++ - GeeksforGeeks.
URL: https://www.geeksforgeeks.org/segmentation-fault-c-cpp/

5. Герасимов В. В., Карпенко Н. В. Проблема невизначеної поведінки коду на мові С при роботі зі структурами // XI Міжнародна науково-практична конференція "Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій", м. Запоріжжя, НУ "Запорізька політехніка", 12-14 грудня 2022 р., с. 140 – 142.

6. Карпенко Н. В., Герасимов В. В. Сучасний підхід до програмування на мові С від нульового до просунутого рівня: навчальний посібник — Дніпро, ПП «Ліра ЛТД», 2022. — 418 с.

7. The Standard – C. URL: https://www.iso-9899.info/wiki/The_Standard

REFERENCES

1. Oshybky i preduprezhdenyia kompyliatora i sredstv sborky C/C++. URL: https://learn.microsoft.com/ru-ru/cpp/error-messages/compiler-errors-1/c-cpp-build-errors?view=msvc-170

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

2. Konfiguratsiya kompilyatora: urovni preduprezhdenij I oshibki URL: https://ravesli.com/konfiguratsiya-kompilyatora-urovni-preduprezhdenij-i-oshibki/
3. Meyers S. Efektivnoe ispolzovanie C++. 55 vernykh sposobov uluchshyt strukturu i kod vashykh programm. – M.: DMK Press, 2006. – 300 s. – ISBN 5-94074-304-8, 0-321-33487-6.
4. Segmentation Fault in C/C++ - GeeksforGeeks.
URL: https://www.geeksforgeeks.org/segmentation-fault-c-cpp/
5. Herasymov V. V., Karpenko N. V. Problema nevyznachenoi povedinky kodu na movi C pry roboti zi strukturamy // XI Mizhnarodna naukovo-praktychna konferentsiia "Suchasni problemy i dosiahnennia v haluzi radiotekhniky, telekomunikatsii ta informatsiinykh tekhnolohii", m. Zaporizhzhia, NU "Zaporizka politekhnika", 12-14 hrudnia 2022 r., s. 140 – 142.
6. Karpenko N. V., Herasymov V. V. Suchasnyi pidkhid do prohramuvannia na movi C vid nulovoho do prosunutoho rivnia: navchalnyi posibnyk — Dnipro, «Lira LTD», 2022. — 418 p.
7. The Standard – C. URL: https://www.iso-9899.info/wiki/The_Standard

Received 12.10.2023.
Accepted 15.10.2023.

Access to struct members and undefined behavior of C code

During software development, novice developers usually receive a lot of error messages and just warnings of various kinds. And if the code simply won't run when there are errors, then the program usually starts when there are various warnings. And here it is important to understand what consequences the presence of warnings of various kinds can lead to.

This work aims to study the code's undefined behavior when working with structures in the C programming language when issuing a corresponding compiler warning about returning the address of a local or temporary variable.

In the procedural programming language C, there is an ancestor of the OOP class — the structure struct, which encapsulates only the state of the entity. And the question arises — is it possible to work with separate components-fields of such a structure analogously to OOP languages?

For research, a simple structure was taken, which contains information about the person's name, surname, and phone number. To access parts of the structure, pseudogetters were used — functions that returned a pointer to the corresponding part of the structure.

The research was conducted in the Visual Studio 2019 environment under the control of the Windows 10 operating system when the default C language standard - MSVC and the more modern ISO standard C17 (2018) was selected in the project settings.

As a result, a truly undefined behavior of the code was obtained, when the result of the work of the code fragment (function call) depends on many factors: the length of the array, the standard of the C language, the position of a certain part in the structure.

An attempt to conduct similar research under the control of the Linux Mint operating system using the gcc compiler version 5.4 was unsuccessful. When compiling the code, a similar warning about returning the address of a local variable was also issued, as in the case of Visual Studio. But when the program was launched, it simply crashed with a message about a segmentation error.

Thus, both the Visual Studio 2019 compiler and the gcc compiler warned us about undefined code behavior. But this uncertain behavior was radically different for operating systems and compilers. If after gcc under the Linux OS, the code simply does not work at all and the program stops its work with a segmentation error message, then after Visual Studio under Windows, inexperienced developers with improper testing and verification of the code can miss the code that "does not always work", which can lead to unexpected results, not always pleasant, to say the least. And that's why software developers, especially beginners, should pay attention not only to compilation errors but also to warnings, even if the code works.

Герасимов Володимир Володимирович - к.т.н, доцент кафедри ЕОМ Дніпровського національного університету імені Олеся Гончара.

Карпенко Надія Валеріївна - к.ф.-м.н., доцент кафедри ЕОМ Дніпровського національного університету імені Олеся Гончара.

Скуратовський Ігор Анатолійович - к.ф.-м.н., доцент кафедри ЕОМ Дніпровського національного університету імені Олеся Гончара.

Gerasimov Volodymyr - Ph.D. in Technical Sciences, Associate Professor, Computer Engineering Department, Oles Honchar Dnipro National University.

Karpenko Nadiia - Ph.D in Physics and Mathematical Sciences, Associate Professor, Computer Engineering Department, Oles Honchar Dnipro National University.

Skuratovskyi Ihor - Ph.D in Physics and Mathematical Sciences, Associate Professor, Computer Engineering Department, Oles Honchar Dnipro National University.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)