Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Реферат

По дисциплине «Основы алгоритмизации и программирования»

На тему «Безусловный оператор goto»

Выполнила:

Студент(ка) 1 курса 7 группы

Подшиваленко Диана Игоревна

Проверил:

Белодед Николай Иванович

2023, Минск

**Содержание**

1. Предисловие … 3
2. Аргументы против goto … 4
3. Аргументы за использование goto … 5
4. Неявное использование goto … 6
5. Примеры правильного использования goto … 7
6. Принципы использования goto … 10
7. Выводы … 11

**Предисловие**

**Оператор goto** — это оператор управления потоком выполнения программ, который заставляет центральный процессор выполнить переход из одного участка кода в другой (осуществить прыжок). Другой участок кода идентифицируется с помощью лейбла(метки).

Оператор goto часто используется в некоторых старых языках, таких как Basic или Fortran, или даже в языке Cи. Однако в C++ goto редко используется, поскольку любой код, написанный с ним, можно более эффективно переписать с использованием других объектов в языке C++, таких как циклы, обработчики исключений или деструкторы.

**1.Аргументы против goto**

Основной аргумент против goto состоит в том, что код без goto — более качественный. Знаменитое письмо Дейкстры «Go To Statement Considered Harmful» («Обоснование пагубности оператора goto») в мартовском номере «Communications of the ACM» 1968 г. положило начало дискуссии. Дейкстра отметил, что качество кода обратно пропорционально количеству goto, использованных программистом. В последующих работах Дейкстра утверждал, что корректность кода, не содержащего goto, доказать легче.

Код с операторами goto трудно форматировать. Для демонстрации логической структуры используются отступы, а goto влияет на логическую структуру. Однако использовать отступы, чтобы показать логику goto и места его перехода, сложно или даже невозможно.

Применение goto препятствует оптимизации, выполняемой компилятором. Некоторые виды оптимизации зависят от порядка выполнения нескольких выражений подряд. Безусловный переход goto усложняет анализ кода и уменьшает возможность оптимизации кода компилятором. Таким образом, даже если применение goto увеличивает эффективность на уровне исходного кода, суммарный эффект из-за невозможности оптимизации может уменьшиться.

Сторонники операторов goto иногда приводят довод, что они делают программу быстрее и проще. Но код, содержащий goto, обычно не самый быстрый и короткий из всех возможных. Изумительная классическая статья Дональда Кнута «Structured Programming with go to Statements» («Структурное программирование и операторы goto») содержит несколько примеров, в которых применение goto приводит к более медленному и объемному коду.

На практике применение операторов goto приводит к нарушению принципа структурного программирования. Если разрешен хотя бы один goto, вместе с пользой в код проникает и вред, так что лучше вообще запретить использование этого оператора, как считают некоторые.

В целом опыт двух десятилетий, прошедших с публикации письма Дейкстры показал всю недальновидность создания кода, перегруженного операторами goto.

В своем обзоре литературы Бен Шнейдерман сделал вывод, что факты свидетельствуют в пользу Дейкстры и нам лучше обходиться без goto, а многие современные языки, включая Java, даже не содержат такой оператор.

**2. Аргументы за использование goto**

Сторонники goto считают, что если осторожно применять оператор при определенных обстоятельствах, то он будет очень даже полезен. Дискуссия о goto вспыхнула, когда Fortran был наиболее популярным языком. Он не имел приличных циклов, и в отсутствие хорошего совета по поводу создания цикла с помощью goto программисты написали кучу спагетти-кода. **Спагетти-код** — это код, порядок выполнения которого напоминает тарелку со спагетти (всё запутано и закручено), что крайне затрудняет следование порядку и понимание логики выполнения такого кода. Если аккуратно использовать goto, то он позволяет заполнить пробел в возможностях, предоставляемых современными языками программирования.

Правильно расположенный goto способен помочь избавиться от дублирования кода. Такой код создает проблемы, если две его части модифицируются по-разному. Дублированный код увеличивает размер исходного и выполняемого файлов. Отрицательный эффект применения goto перевешивается недостатками дублированного кода.

Оператор goto может пригодиться в методе, который сначала распределяет ресурсы, выполняет с ними какие-то операции, а потом освобождает эти ресурсы. Используя goto, вы можете выполнять очистку в одном месте. Оператор goto уменьшает вероятность того, что вы забудете освободить ресурсы при обнаружении ошибки.

Порой goto позволяет создать более быстрый и короткий код.

Хорошее программирование не означает исключение всех goto. Систематическая декомпозиция, усовершенствование и разумный выбор управляющих структур обычно автоматически приводит к программам, не содержащим goto. Стремление к коду без goto — это не цель, а результат, и бесполезно заострять внимание исключительно на устранении goto.

Десятилетия исследований операторов goto не смогли продемонстрировать их вредоносность.

И, наконец, операторы goto входят во множество современных языков, включая Visual Basic и Ada — наиболее тщательно продуманный язык программирования в истории. Ada создавался уже после того, как были приведены все аргументы с обеих сторон дискуссии по goto, и после всестороннего рассмотрения вопроса разработчики Ada решили включить в него goto.

**3.Неявное использование goto**

В языке С есть много операторов, которые на самом деле являются банальным goto – условным или безусловным. Это все виды циклов for (…), while (…) {…}, do {…} while (…). Это анализ числовых переменных switch (…) {case … case …}. Это те же операторы прерывания/перехода в циклах break и continue. В конце концов, это вызовы функций и выход из них return. С++ добавляет массу невидимых переходов перегрузками, конструкторами и деструкторами. Из явных переходов – try {… throw …} catch (…) {…}.

**4. Примеры правильного использования goto**

1. Выход из вложенных циклов:

#include <iostream>

#include <Windows.h>

#include <ctype.h>

#include <cstring>

#include <iomanip>

#include <fstream>

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int table[4][4] = { {1, 3, 5, 7}, {2, 4, 6, 8}, {9, 11, 13, 15}, {10, 12, 14, 16}};

int var = 11;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (table[i][j] == var) {

goto end;

}

}

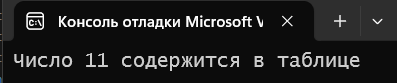
}

cout << "Число 11 не содержится в таблице" << endl;

return 0;

end: cout << "Число 11 содержится в таблице" << endl;

}

Результат выполнения:

Да, этот же код можно реализовать без goto, однако нам придётся добавить переменные-флаги, а также после каждого прохода внутреннего цикла делать проверку флага и дополнительно по окончанию внешнего цикла(и по сути мы используем неявный goto в виде break).

#include <iostream>

#include <Windows.h>

#include <ctype.h>

#include <cstring>

#include <iomanip>

#include <fstream>

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int table[4][4] = { {1, 3, 5, 7}, {2, 4, 6, 8}, {9, 11, 13, 15}, {10, 12, 14, 16}};

int var = 11;

bool t = false;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (table[i][j] == var) {

t = true;

break;

}

}

if (t) {

cout << "Число 11 содержится в таблице" << endl;

break;

}

}

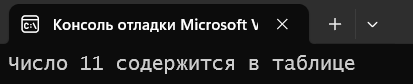
if (!t) {

cout << "Число 11 не содержится в таблице" << endl;

}

return 0;

}

Результат выполнения:

2. Обработка ошибок:

#include <iostream>

#include <Windows.h>

#include <ctype.h>

#include <cstring>

#include <iomanip>

#include <fstream>

using namespace std;

char openfile(const char\* i) {

FILE\* my\_file = 0;

if (!(my\_file = fopen(i, "r"))) {

goto not\_open;

}

return 'o';

not\_open:

return 'e';

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const char\* filename = "test.txt";

char result = openfile(filename);

switch (result) {

case 'e':{

cout << "Ошибка открытия" << endl;

break;

}

case 'o':{

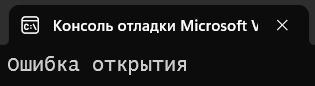
cout << "Файл успешно открыт" << endl;

break;

}

}

}

Если файл не найден, то:

**5.Принципы использования goto**

**1.** Применяйте goto для эмуляции структурированных управляющих конструкций в языках, не поддерживающих их напрямую. Причем эмулируйте их точно — не злоупотребляйте дополнительной гибкостью, предоставляемой оператором goto.

**2.** Не используйте goto, если можно заменить эквивалентной встроенной конструкцией.

**3.** Измеряйте производительность всех goto, используемых для повышения эффективности. В большинстве случаев вы можете переписать код без goto с целью повышения читабельности и при этом не потерять в эффективности.

**4.** Ограничьте использование меток goto на метод, если только вы не эмулируете управляющие конструкции.

**5.** Используйте операторы goto так, чтобы их переходы были только вперед, а не назад, если только вы не эмулируете управляющие конструкции.

**6.** Убедитесь, что используются все метки goto. Неиспользуемые метки могут служить признаком недописанного кода, а именно того, в котором осуществляется переход по этим меткам. Если метки не используются, удалите их.

**7.** Убедитесь, что goto не приводит к созданию недостижимого кода.

**Выводы**

Использовать или не использовать goto – это вопрос каждого. В современных языках программирования вы легко можете заменить девять из десяти операторов goto эквивалентными последовательными конструкциями. В этих простых случаях вы должны заменять операторы goto просто по привычке. В сложных ситуациях вы также можете не использовать goto в девяти случаях из десяти. Например, можно разбить код на меньшие по размеру методы, использовать обработчики исключений try-finally или вложенные if, проверять и перепроверять статусную переменную или реструктурировать условные выражения.

**Плюсы использования goto:**

1. Один из самых оптимальных выходов из нескольких вложенных циклов и switch… case.

2. В Си: наиболее экономичный способ обработки ошибок.

3. В отдельно взятых случаях можно построить самый оптимальный алгоритм.

4. Экономит память при аккуратном использовании, что иногда бывает очень важно

**Минусы использования goto:**

1. Непривычность кода и нарушение стандартного выполнения программы(сверху-вниз), получается так называемый “спагетти-код”.

2. Усложнение компилятору процесса оптимизации кода.

3. Повышение вероятности появления труднонаходимых ошибок в коде.