ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ТАДЖИКИСТАНА
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИСКУССТВЕННОГО ИНТЕЛЛЕКТА
КАФЕДРА ПРОГРАММИРОВАНИЯ И КОМПЬЮТЕРНОЙ ИНЖЕНЕРИИ

К У Р С О В А Я Р А Б О Т А

по дисциплине «Объектно – ориентированное программирование»
Тема: «Анализ применения условных операторов в разных языках программирования»

Выполнил(а): _____ студент(ка) 2–го курса
спец. 1-40010103 В
Кадыров Абуали
Абдугаффорович

Руководитель: _____ ст. преп. Шарипов Ш.А.

Душанбе – 2023

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ТАДЖИКИСТАНА

ФАКУЛЬТЕТ ИФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

КАФЕДРА ПРОГРАММИРОВАНИЯ И КОМПЬЮТЕРНОЙ ИНЖЕНЕРИИ

«УТВЕРЖДАЮ»
Заведующий кафедрой
программирования
и компьютерной инженерии
_____ к.т.н., и.о. доц. Гуломсафдаров А.Г.
«____» _____ 2023 г.

ЗАДАНИЕ
на курсовую работу

Ф.И.О. студента: Кадыров Абуали Абдугаффорович.
Специальность: *1–40010103 В «*Компьютерные и банковские системы»
Тема работы: «Анализ применения условных операторов в разных языках программирования»
Утверждено на заседании кафедры: *протокол № 7 от 23–го февраля 2023 г.*

1. СОДЕРЖАНИЕ КУРСОВОЙ РАБОТЫ

№ п/п	Наименование разделов	Объем в %	Срок выполнения
	Содержание	3	01.03.2023
	Введение	10	08.03.2023
1.	Глава 1: Исследование особенностей организации системы ветвлений в языке Программирования с++	15	13.03.2023
2.	Глава 2: Описания операторов	20	10.04.2023
	Заключение	10	11.04.2023
	Список литературы	5	17.04.2023
	Оформление курсовой работы	5	9.05.2023

2. ЛИТЕРАТУРА

1. 1. Campbell Parallel Programming with Microsoft® Visual C++® / Campbell. – Москва: Гостехиздат, 2011. – 784 с.

2. 2. Альфред, В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. – Москва: Высшая школа, 2015. – 882 с.

3. 3. Балена, Франческо Современная практика программирования на Microsoft Visual Basic и Visual C# / Франческо Балена , Джузеппе Димауро. – М.: Русская Редакция, 2015. – 640 с.

3. ПРОЦЕНТОВКА

1. 13.03.2023 **до** 20.03.2023 **2.** 17.04.2023 **до** 24.04.2023

Курсовая работа принял(а) к выполнению: _____ Кадыров А.А
(подпись)

Дата защиты курсовой работы «_____»

Руководитель: _____
Шарипов Ш.А.

(подпись)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одной из проблем, с которой сталкивается программист при написании параллельной программы, является семантический разрыв между параллельным алгоритмом в его математической форме и его реализацией на многопроцессорной вычислительной системе (МВС). Структурно-процедурная организация вычислений позволяет решать эту проблему. Для эффективной реализации на многопроцессорных системах вычислительных алгоритмов необходим язык программирования высокого уровня (ЯПВУ), обладающий следующими качествами.

Язык должен эффективно реализовываться на различных МВС;

Иметь средства для записи алгоритмов без изменения их начальной параллельной структуры;

Язык должен позволять программисту максимально просто описывать различные виды параллелизма, реализованные в МВС, в достаточно сжатом виде;

Содержать мощные конструкции, как в традиционных ЯПВУ типа Фортрана, Си и т.п.

Для МВС со структурно-процедурной организацией вычислений (СПОВ) возможен качественно новый принцип описания задачи: задача делится на независимые, функционально законченные участки, реализуемые структурно (аппаратно) на модулях с ПЛИС-архитектурой; смена вычислительных структур происходит под управлением программы. Таким образом, в ходе выполнения программы может выполняться только одна вычислительная конструкция.

Значительную сложность в ЯПВУ многопроцессорных систем представляет проблема, связанная с программированием связей между компонентами системы. Поскольку МВС СПОВ имеет полнодоступную систему коммутации, можно переложить решение этой задачи на транслятор. Таким образом, программист будет пользоваться не только виртуальной памятью, но и виртуальной системой коммутации.

1.ИССЛЕДОВАНИЕ ОСОБЕННОСТЕЙ ОРГАНИЗАЦИИ СИСТЕМЫ ВЕТВЛЕНИЙ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ С++.

Анализ базового уровня данной системы.

В самом простом случае компьютерная программа выполняется по порядку, строка за строкой. Однако в подавляющем большинстве случаев на каком-то этапе выполнения потребуется изменить ход выполнения программы в зависимости от результатов, полученных в предыдущих выражениях. В таком случае необходимо проверить выполнение некоторых условий, и изменить ход выполнения программы в соответствии с ними. Для программирования ветвлений в языке С++ имеются инструкции if…else и switch, а также тернарная операция (?:)..

1.1 Инструкция выбора if…else

Выполняет указанное в ней действие, только если условие истинно и пропускает его в ином случае. Инструкция (структура) if…else позволяет определить различные действия, которые должны выполняться в случаях, если условие истинно или ложно. Допускается использование вложенных инструкций if. Формат инструкции if имеет вид

If (логическое выражение) выражение-1; else выражение-2;
Или
If (логическое выражение) выражение-1;

При этом выражение-1 и выражение-2 могут быть составными (т.е. представлять собой совокупность последовательно выполняемых выражений, заключенных в операторные скобки), или в более общем случае –

структурированными. Отметим также, что вложенная структура if…else может работать гораздо быстрее, чем серия структур if с единственным выбором вследствие возможности быстрого выхода после удовлетворения одному из условий.

При построении логических выражений участвуют логические операторы, наиболее часто используются операторы (&&), (||), (!). Операторы (&&) и (||) вычисляют второй аргумент только в случае, если в этом есть необходимость. Так, например, в выражении

If (a>b && c>100) a=c;

Вначале проверяется первое условие a>b, а проверка условия c>100 будет осуществляться только в том случае, если первое условие истинно. В выражении

If (a>b || c!=0) a=c;

Второе условие будет проверяться только в случае, если первое условие ложно; если же первое условие истинно, то второе условие не проверяется, и результатом проверки сразу возвращается true. Такая реализация проверок носит название механизма быстрых вычислений.

При написании условий можно объявлять переменную прямо в условии. В этом случае область видимости такой переменной будет простираться от точки ее объявления до конца инструкции, контролируемой условием. Если в инструкции if есть ветвь else, область видимости такой переменной простирается на обе ветви. При этом объявление в условии должно объявлять и инициализировать единственную переменную или константу.

Такой подход к объявлению переменной приводит к более компактному коду программы. Еще одним преимуществом такого объявления (когда такое возможно) является то, что если объявить переменную до инструкции if, то

время ее жизни будет простираться до конца блока, в котором она была объявлена, даже если она не будет использована после проверки условия.

Ниже в качестве примера приводится текст программы, выводящей на экран значение корня из указанного числа, если число является полным квадратом, и «No» в противном случае.

```
#include
#include
#include
Using namespace std;
Long f (long a)
{long s=pow(a,0.5);
If (s*s == a) return s; else return 0;}
Int main()
{ long ch;
Cout<<»ch= «; cin>>ch;
If (long sum = f(ch)) cout<<»Yes: «< else cout<<»No»;
Getch(); return 0;}
```

Ясно, что для решения такой простой задачи нет смысла писать отдельную функцию, -- в данном случае это сделано лишь для демонстрации возможности объявления переменной в условии.

Условная операция ?:

В C++ имеется еще условная операция (?:) – единственная тернарная операция в C++ с тремя операндами, которая близка к структуре if…else. Первый операнд является условием, второй – содержит значение условного выражения в случае, если условие истинно, а третий операнд равен значению условного выражения, если условие ложно. Например, оператор вывода

Cout << (a >=10 ? «Да» : «Нет»);

Содержит условное выражение, равное строке «Да», если условие a>=10 истинно, и равно строке «Нет», если оно ложно. Т.к. условная операция имеет низкое старшинство, в приведенном выражении потребовались скобки.

Значения условного выражения могут быть также некоторыми исполняемыми действиями.

Например, условное выражение

A >=10 ? cout << «Да \n» : cout << «Нет \n»;

Следует понимать так: «Если значение переменной а больше или равно 10, то выдать строку «Да», иначе строку «Нет»».

1.2 Инструкция switch (структура множественного выбора)

Иногда алгоритм решения какой-либо задачи содержит ряд альтернатив, причем некоторую переменную надо проверять отдельно для каждого постоянного целого значения, которое она может принимать. В зависимости от результатов этой проверки должны выполняться различные действия. Ветвление программирование switch

Инструкция switch состоит из ряда меток case и необязательной метки default. Структура инструкции имеет вид:

Switch (a) {
Case a1: выражение-1; break;
Case a2: выражение-2; break;
……;
Case aN: выражение-N; break;
Default: выражение-0; break; }

Здесь a – выражение любого целого типа (в том числе и символьного), значение которого вычисляется; a1, a2, …, aN – константы, с которыми сравнивается значение выражения a; выражение-0 – выражение, которое выполняется, если значение выражения a не совпадает ни с одной из констант a1, a2, …, aN.

Инструкция break вызывает передачу программного управления на первое выражение после структуры switch. В случае если инструкцию break не указывать, то условия case в инструкции switch работают совместно: если везде в приведенной выше структуре switch не использовать break, тогда каждый раз, когда одно из условий caseудовлетворяется, будут выполняться выражения всех последующих меток case. Если ни одно из условий не выполнено, то выполняются выражения после метки default (если в структуре switch метка default помещена последней в списке, то инструкция break в ней не требуется).

Отметим важную особенность структуры switch, отличающую ее от всех других структур: при нескольких действиях после case не требуется заключать их в фигурные скобки.

Как было отмечено выше, важной особенностью языка C++ является то, что символы могут храниться в любом целом типе данных, поскольку они представляются как однобайтовое целое. Это означает, что можно трактовать символ или как целое, или как символ, в зависимости от его использования. Например, выражение

Cout<<»Символ (a) в ASCII имеет значение «< ('a');

Напечатает символ a и его целочисленное представление в таблице ASCII (American Standard Code for Information Interchange) в виде:

Символ (a) в ASCII имеет значение 97

Поэтому структура switch может применяться с любой комбинацией символьных и целых констант, имеющих целое постоянное значение. Отметим также, что в теле инструкции switch можно использовать вложенные инструкции switch, при этом в ключевых словах case можно использовать одинаковые константные выражения. Однако такие вложенные структуры обычно встречаются в программах очень редко.

Пример использования инструкции switch приведен ниже.

Задача

Дан равносторонний треугольник. Будем считать, что может задаваться лишь один из его параметров: сторона, площадь, высота, радиус вписанной окружности или радиус описанной окружности. Составить программу, которая, в зависимости от того, какой из параметров указан, будет на его основе находить остальные.

```
#include
#include
#include
Using namespace std;
Int main()
{ float a,h,S; //сторона, высота и площадь треугольника
Float r,R; //радиус вписанной и описанной окружностей
Int variant; //номер сделанного выбора (вариант выбора)
System(«cls»);
Cout<<»Выберите известный параметр из списка: «<>variant;
System(«cls»);
Switch (variant) {
Case 1: cout<<»Введите сторону треугольника»<>a;
Cout<<»Высота треугольника h=»
<>h;
Cout<<»Сторона треугольника a=»<<2*h/sqrt(3)<>S;
Cout<<»Сторона треугольника a=»
<>r;
Cout<<»Сторона треугольника a=»<<2*r*sqrt(3)<>R;
Cout<<»Сторона треугольника a=»<
```

Инструкция goto (безусловного перехода)

Инструкция goto позволяет изменить стандартный последовательный порядок выполнения выражений и перейти к выполнению программы, начиная с заданного выражения. Выражение, на которое происходит переход, должно быть помечен меткой (описывать метки не требуется).

Одной меткой можно пометить только одно выражение. Метка от помеченного выражения отделяется двоеточием. Имя метки – это идентификатор. Формат инструкции следующий:

Goto имя-метки;
……..;
Имя-метки: выражение;

Используя инструкцию goto, можно передавать управление внутрь составного выражения. Однако при входе в составное выражение нужно соблюдать осторожность, т.к. оно может дать неправильный результат. Например, в составном выражении, содержащем объявления переменных с инициализацией, так как объявления располагаются перед выполняемыми выражениями, значения объявленных переменных при таком переходе будут не определены. Кроме того, безусловный переход можно осуществлять далеко не из каждого места программы и не в любое ее место.

2.ОПИСАНИЯ ОПЕРАТОРОВ

Пустой оператор

Пустой оператор состоит только из точки с запятой. При выполнении этого оператора ничего не происходит. Он обычно используется в следующих случаях:

- в операторах do, for, while, if в строках, когда место оператора не требуется, но по синтаксису требуется хотя бы один оператор;

- при необходимости пометить фигурную скобку.

Синтаксис языка С требует, чтобы после метки обязательно следовал оператор. Фигурная же скобка оператором не является. Поэтому, если надо передать управление на фигурную скобку, необходимо использовать пустой оператор.

Пример:

```
Int main ( )
{
```

```
;
{
If (...) goto a; /* переход на скобку */
{ ...
}
A:;
}
Return 0;
}
```

Оператор безусловного перехода

Использование оператора безусловного перехода goto в практике программирования на языке С настоятельно не рекомендуется, так как он затрудняет понимание программ и возможность их модификаций.

Формат этого оператора следующий:

```
Goto имя-метки;
…
Имя-метки: оператор;
```

Оператор goto передает управление на оператор, помеченный меткой имя-метки. Помеченный оператор должен находиться в той же функции, что и оператор goto, а используемая метка должна быть уникальной, т.е. одно имя-метки не может быть использовано для разных операторов программы. Имя-метки – это идентификатор.

Любой оператор в составном операторе может иметь свою метку. Используя оператор goto, можно передавать управление внутрь составного оператора. Но нужно быть осторожным при входе в составной оператор, содержащий объявления переменных с инициализацией, так как объявления располагаются перед выполняемыми операторами и значения объявленных переменных при таком переходе будут не определены.

2.1 Операторы условного перехода

Оператор if

Формат оператора:

If (выражение) оператор;

Выполнение оператора if начинается с вычисления выражения.

Далее выполнение осуществляется по следующей схеме:

- если выражение истинно (т.е. отлично от 0), то выполняется оператор.

- если выражение ложно, то выполняется следующий за if оператор.

После выполнения оператора if значение передается на следующий оператор программы, если последовательность выполнения операторов программы не будет принудительно нарушена использованием операторов перехода.

Пример:

if (i < j) i++:

Оператор if-else

Формат оператора:

If (выражение) оператор-1; else оператор-2;

Выполнение оператора if начинается с вычисления выражения.

Далее выполнение осуществляется по следующей схеме:

- если выражение истинно (т.е. отлично от 0), то выполняется оператор-1.

- если выражение ложно (т.е. равно 0),то выполняется оператор-2.

После выполнения оператора if значение передается на следующий оператор программы, если последовательность выполнения операторов программы не будет принудительно нарушена использованием операторов перехода.

Пример:

If (i < j) i++:

Else { j = i-3; i++; }

Этот пример иллюстрирует также и тот факт, что на месте оператор-1, так же как и на месте оператор-2 могут находиться сложные конструкции.

Допускается использование вложенных операторов if. Оператор if может быть включен в конструкцию if или в конструкцию else другого оператора if. Чтобы сделать программу более читабельной, рекомендуется группировать операторы и конструкции во вложенных операторах if, используя фигурные скобки. Если же фигурные скобки опущены, то компилятор связывает каждое ключевое слово else с наиболее близким if, для которого нет else.

Примеры:

```
Int main ( )
{
Int t=2, b=7, r=3;
If (t>b)
{
If (b < r) r=b;
}
Else r=t;
Return (0);
}
```

В результате выполнения этой программы r станет равным 2.

Если же в программе опустить фигурные скобки, стоящие после оператора if, то программа будет иметь следующий вид:

```
Int main ( )
{
Int t=2,b=7,r=3;
If ( a>b )
If ( b < c ) t=b;
Else
R=t;
Return (0);
}
```

В этом случае r получит значение равное 3, так как ключевое слово else относится ко второму оператору if, который не выполняется, поскольку не выполняется условие, проверяемое в первом операторе if.

Оператор if-else if

Следующий фрагмент иллюстрирует вложенные операторы if:

```
Char ZNAC;
Int x,y,z;
:
If (ZNAC == '-') x = y – z;
Else if (ZNAC == '+') x = y + z;
Else if (ZNAC == '*') x = y * z;
Else if (ZNAC == '/') x = y / z;
Else …
```

Из рассмотрения этого примера можно сделать вывод, что конструкции использующие вложенные операторы if, являются довольно громоздкими и не всегда достаточно надежными. Другим способом организации выбора из множества различных вариантов является использование специального оператора выбора switch.

2.2 Оператор переключатель switch

Оператор switch предназначен для организации выбора из множества различных вариантов. Формат оператора следующий:

```
Switch ( выражение )
{ [объявление]
:
[ case константное-выражение1]: [ список-операторов1]
[ case константное-выражение2]: [ список-операторов2]
:
:
[ default: [ список операторов ]]
```

}

Выражение, следующее за ключевым словом switch в круглых скобках, может быть любым выражением, допустимыми в языке С, значение которого должно быть целым.

Значение этого выражения является ключевым для выбора из нескольких вариантов. Тело оператора smitch состоит из нескольких операторов, помеченных ключевым словом case с последующим константным-выражением. Следует отметить, что использование целого константного выражения является существенным недостатком, присущим рассмотренному оператору.

Так как константное выражение вычисляется во время трансляции, оно не может содержать переменные или вызовы функций. Обычно в качестве константного выражения используются целые или символьные константы.

Все константные выражения в операторе switch должны быть уникальны. Кроме операторов, помеченных ключевым словом case, может быть, но обязательно один, фрагмент, помеченный ключевым словом default.

Список операторов может быть пустым, либо содержать один или более операторов. Причем в операторе switch не требуется заключать последовательность операторов в фигурные скобки.

Отметим также, что в операторе switch можно использовать свои локальные переменные, объявления которых находятся перед первым ключевым словом case, однако в объявлениях не должна использоваться инициализация.

Схема выполнения оператора switch следующая:

- вычисляется выражение в круглых скобках;

- вычисленные значения последовательно сравниваются с константными выражениями, следующими за ключевыми словами case;

- если одно из константных выражений совпадает со значением выражения, то управление передается на оператор, помеченный соответствующим ключевым словом case;

- если ни одно из константных выражений не равно выражению, то управление передается на оператор, помеченный ключевым словом default, а в случае его отсутствия управление передается на следующий после switch оператор.

Отметим интересную особенность использования оператора switch: конструкция со словом default может быть не последней в теле оператора switch. Ключевые слова case и default в теле оператора switch существенны только при начальной проверке, когда определяется начальная точка выполнения тела оператора switch. Все операторы, между начальным оператором и концом тела, выполняются вне зависимости от ключевых слов, если только какой-то из операторов не передаст управления из тела оператора switch. Таким образом, программист должен сам позаботится о выходе из case, если это необходимо. Чаще всего для этого используется оператор break.

Для того, чтобы выполнить одни и те же действия для различных значений выражения, можно пометить один и тот же оператор несколькими ключевыми словами case.

Пример:

```
Int i=2;
Switch(i)
{
case 1: i+=2;
case 2: i*=3;
case 4: i/=2;
case 0: i-=5;
default ;
}
```

Выполнение оператора switch начинается с оператора, помеченного case 2. Таким образом, переменная i получает значение, равное 6, далее выполняется оператор, помеченный ключевым словом case 0, а затем case 4, переменная i

примет значение 3, а затем значение -2. Оператор, помеченный ключевым словом default, не изменяет значения переменной.

Рассмотрим ранее приведенный пример, в котором иллюстрировалось использование вложенных операторов if, переписанной теперь с использованием оператора switch

```
char ZNAC;
int x, y,z;
switch (ZNAC){
case'+'x=y+z;break;
case'-'x=y-z;break
case'*'x=y*z;break
case'/'x=y/z;break
default:;
}
```

Использование оператора break позволяет в необходимый момент прервать последовательность выполняемых операторов в теле оператора switch, путем передачи управления оператору, следующему за switch

Отметим, что в теле оператора switch можно использовать вложенные операторы switch, при этом в ключевых словах case можно использовать одинаковые константные выражения.

Пример:

```
Switch (a) {
case 1: b=c; break;
case 2: }
switch (d) {
case 0: f=s; break;
case 1: f=9; break;
case 2: f-=9; break;
 }
Case 3: b-=c; break;
```

:

}

ЗАКЛЮЧЕНИЕ

Типом данных называется множество допустимых значений этих данных, а также совокупность операций над. Типы делятся на следующие группы:

простые, структурированные, указатели, процедурные, объекты. Есть стандартные (предопределенные) и определяемые программистами в разделе, начинающемся со слова Type. Простые типы определяют упорядоченное множество значений элементов и делятся на вещественные, целые, символьный, логический, перечисляемый тип-диапазон. Вещественные типы определяют дробные числа и представлены 5 стандартными типами: real, single, double, extended, comp. Целые типы определяют целые числа и представлены стандартными типами: integer, longint, shortint, byte, word, стандартный символьный тип char определяет полный набор допустимых символов. Стандартный логический тип Boolean представляет собой тип данных, каждый элемент которого может принимать 1 из 2-х значений: False (ложь), True (правда). Перечисляемый тип не является стандартным и определяется набором идентификаторов, к которым может совпадать значение элемента данных.

В заключение можно сказать, что условные операторы – это необходимый инструмент в программировании, который позволяет разработчикам создавать более гибкий и адаптивный код. Благодаря условным операторам в языках программирования» можно создавать многофункциональные программы, которые могут изменять свое поведение в зависимости от входных данных и условий.

Язык высокого уровня для МВС со структурно-процедурной организацией вычислений позволяет эффективно реализовать любой тип параллельной обработки. Использование неявного описания параллелизма, виртуальных систем коммутации и распределенной памяти упрощает программирование. Язык отражает все особенности структурно-процедурной организации вычислительного процесса и позволяет оперативно разрабатывать прикладное программное обеспечение.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Керниган Б., Ритчи Д., Фьюэр А. Язык программирования Си: Задачи по языку Си. М.: Финансы и статистика, 1985. – 192с.
2. Керниган Б., Ритчи Д. Язык программирования Си. М.: Финансы и статистика, 1992. – 272с.
3. Подбельский В.В., Фомин С.С. Программирование на языке Си. Учеб. Пособие. М.: Финансы и статистика, 2004. 600 с
4. .Подбельский В.В. Язык Cu ++: Учебное пособие. – М.: Финансы и статистика,1995, -560 с.
5. Страуструп Б. Язык программирования Сг ++. – М.: Радио и связь, 1991. – 352 стр.
6. Собоцинский В.В. Практический курс Turbo Cu ++. Основы объктно-ориентированного программирования. – М.: Свет, 1993. – 236 с.
7. Романов В.Ю. Программирование на языке Cu ++. Практический подход. – М.: Компьтер, 1993. – 160 с.
8. Уинер Р. Язык турбо Cu . – М.: Мир, 1991. – 384 с.
9. Юлин В.А., Булатова И.Р. Приглашение к Cu. – Мн.: Высш. Шк., 1990,-224 с.
10. Котлинская Г.П., Галиновский О.И. Программирование на языке Cu. - Мн.: Высш. Шк., 1991. – 156 с.