Содержание:

ВВЕДЕНИЕ

Актуальность темы исследования. Для того чтобы программа обладала универсальностью, действия в ней должны совершаться не над постоянными, а над переменными величинами. Поэтому важным понятием программирования является понятие переменной.

Любая программа, даже написанная на другом языке программирования, в основном состоит из обработки некоторых значений, которые в свою очередь хранятся в переменных.

Переменные используются в любом языке программирования. Чтобы использовать переменную в программе, ее необходимо объявить. Хотя все данные на компьютере, в сущности, имеют одинаковый вид (наборы нулей и единиц), переменные бывают разных видов, называемых в их случае типами.

Чтобы использовать переменные, их нужно объявлять. Это означает, что им необходимо назначать имя и указать тип. После объявления их можно начинать использовать в качестве единиц хранения для данных

Если говорить простым языком, то переменная - это небольшая именованная "конструкция" содержащая значение любого типа. Переменные - это самая лёгкая часть любого языка с которой всегда следует начинать изучение переходя далее, к более сложным структурам языка.

Цель работы - дать понятие переменной в программировании, рассмотреть виды и типы переменных.

В соответствии с поставленными целями решались следующие основные **задачи**:

- дать понятие и рассмотреть характеристики переменной в программировании;

- изучить свойства переменной;

- рассмотреть типы данных C++;

- изучить особенности объявления переменных в C++.

Методы исследования:

- обработка, анализ научных источников;

- анализ научной литературы, учебников и пособий, а также сети Интернет по исследуемой проблеме.

Объект исследования – переменные в программировании

Предмет исследования – виды и типы переменных в программировании

1. СУЩНОСТЬ И СВОЙСТВО ПЕРЕМЕННЫХ В ПРОГРАММИРОВАНИИ

1.1 Понятие и характеристики переменной в программировании

Под переменной в языках программирования понимают программный объект (число, слово, часть слова, несколько слов, символы), имеющий имя и значение, которое может быть получено и изменено программой.

В программировании переменная (variable) это своего рода емкость для хранения данных. Когда информация записана в переменной (или по-другому, когда переменной присвоено значение), тогда эту информацию можно изменять, выводить в окне Web-браузера, посылать по электронной почте и т.д.

Переменная гибка:

- в ней можно хранить информацию;
- можно из неё извлекать информацию, что не повлияет на значение самой переменной;
- в неё можно записать новые данные.

Причем, эти действия можно выполнять практически сколько угодно раз. Из названия ясно, что переменная – вещь непостоянная. Например, в языке программирования PHP переменные существуют или содержат в себе значение исключительно во время работы скрипта. Как только завершается исполнение

скрипта, так и существование переменных прекращается. То есть при переходе с одной страницы web-сайта на другую, переменных и их значений прежней страницы уже нет. Это кроме случаев, когда нужные значения специально передаются другой странице.

Переменные появились с первыми языками программирования. Результат работы любой программы сводится к выполнению действий над какими-либо данными. Напомним, что память (memory) – это последовательность байтов (byte), каждый из которых принимает значения от 0 до 255. Так как байтов неимоверно много, единственный способ различать их - это присвоение каждому из них порядкового номера. Так и есть. Каждый байт оперативной памяти доступен процессору через его порядковый номер. Этот порядковый номер называется адресом байта[1].

Во времена, когда программы писались на машинном коде, программист должен был запоминать в какой участок памяти он записал нужное значение. Представьте, как усложнялся процесс написания программы, когда возникала необходимость работы с несколькими значениями. Адрес байта памяти есть число, которое мало о чем говорит. Большой объем памяти создает трудности программисту.

С первыми языками программирования появилась полезная возможность связывания определенного участка оперативной памяти с символьным названием (набором символов). По сравнению с адресом название переменной может отражать содержимое этого участка памяти. Но имя переменной не единственная вещь, которая определяет переменную. Процессор может обрабатывать три вида данных: байт, слово и двойное слово. (?) (Термины «слово» и «двойное слово» здесь используются в узкоспециальном смысле, отражая собой размер участка памяти, выделенного под переменную.) Поэтому определение вида переменной в языках нижнего и среднего уровней происходит обычно указанием типа переменной. Эти два свойства переменной (название и тип) определяют нужный участок памяти и способ его использования. В большинстве случаев именно тип переменной определяет сколько байтов памяти захватит переменная. Например, переменной типа BYTE присвоено имя A. Процессор по названию переменной (A) определяет её место в памяти и при извлечении значения этой переменной воспользуется командой, предназначенной для извлечения байта (не слова и не двойного слова).

В общем случае, переменная – это поименованный участок оперативной памяти, используемый для временного хранения данных. В зависимости от языка программирования, объявление переменной может сопровождаться указанием его типа. Обычно в языках высокого уровня тип не указывается. Синтаксические

различия между языками проявляются как раз в объявлении переменных. В C и C++ необходимо указывать тип, в PHP используется префикс $.

В современном мире программирования программист должен знать не только имя и тип переменной. Также существуют понятия пространства имен и область действия переменной. Представьте, что создается программа, в которой используются несколько переменных. Имена этих переменных составляют список, который определяет пространство имен. Представим, что в ходе создания программы мы, по ошибке, объявили две переменные с одинаковыми названиями. При попытке запуска программы его компилятор сообщит об этой ошибке. Это было бы невозможно, если бы компилятор не контролировал переменные. То есть контроль безупречности пространства имен возлагается от программиста на компилятор; что облегчает процесс создания и отладки программы. На практике, приведенный пример не во всех языках приводит к ошибке. Каждый компилятор (или интерпретатор) имеют собственные требования к пространству имен. То, что является ошибкой в одном языке (в данном случае C или C++), в других языках ошибкой может не быть.

Если раньше программы были небольшие и их исходный код располагался в одном файле, то сейчас текст кода может состоять из нескольких файлов. И запомнить уникальное имя каждой переменной, использующейся в программе, становится практически невозможным. Поэтому (и не только) было введено понятие области действия (или области существования) переменных. Область действия – понятие абстрактное. Это понятие применяется только в языках среднего и высокого уровней. Целью применения области действия является разделение пространства имен на несколько независимых частей. Так в программе могут существовать несколько переменных с одинаковыми именами и типами, не мешаю друг другу. Начинающему идея разделения области действия в пределах отдельных файлов исходного кода программы должна быть более понятна. Например, этот подход используется в PHP. Это означает, что переменная, объявленная в одном файле, может быть абсолютно недоступна в других файлах. Например, можем объявить переменную с именем MyVar в нескольких файлах проекта, и это не будет ошибкой [2].

Таким образом, переменная в программировании обладает следующими характеристиками:

1. имя
2. адрес

3. тип
4. размер, который обычно определяется типом
5. принадлежность какому-либо пространству имен
6. область действия.

1.2 Свойства переменной

Свойством переменной является её способность получать некоторое значение от программы, удерживать его в течение времени работы программы и сообщать это значение программе при запросах программы. Для каждой переменной в программе компьютер отводит часть памяти необходимого размера. Обратимся к рис. 1.1 и проследим, как устроены переменные.

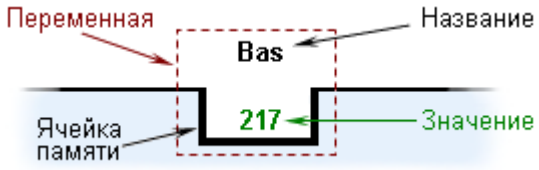

Рис. 1.1. Переменная в памяти компьютера.

В ячейке памяти компьютера содержится значение переменной. Это значение может быть затребовано для обработки и изменено программой. Имя переменной никогда не изменяется. При создании кода программист может назначить переменной любое имя, но с момента, когда готовая программа запущена в работу, ни у программиста, ни у компьютера, ни у программы нет технической возможности изменить название переменной.

Если на пути выполнения программы встречается имя переменной, то программа обращается к этой переменной, чтобы получить для обработки её значение. Если программа обратилась к переменной, последняя сообщает программе своё значение. При этом значение переменной остаётся неизменным. Программа же получает в своё распоряжение копию значения, содержащегося в ячейке памяти, выделенной для этой переменной (рис. 1.2).

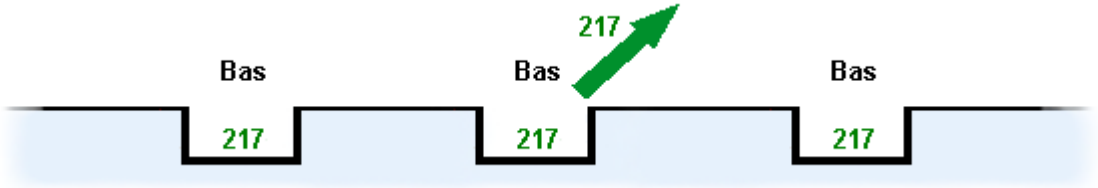

Рис. 1.2. Состояние ячейки памяти переменной при сообщении значения

программе

При сообщении значения переменной программе оно остаётся неизменным. Название (имя) переменной никогда не изменяется.

В течение некоторого времени переменная не имеет отношений с исполняемой программой. В это время программа может обращаться к другим переменным или производить необходимые вычисления. В период между случаями общения с программой переменная удерживает своё значение, т.е. сохраняет его неизменным [3].

В соответствии с выполняющимся алгоритмом, заложенным в программу, может потребоваться изменение значения переменной. В этом случае программа сообщает переменной её новое значение, а переменная получает это значение от программы. При этом в ячейке памяти производятся необходимые преобразования, в результате которых предыдущее значение переменной удаляется, а на его месте образуется новое значение переменной, сообщённое программой (рис. 1.3).

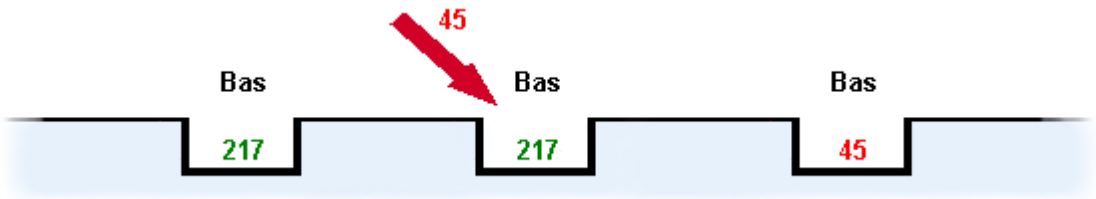

Рис. 1.3 Состояние ячейки памяти переменной при получении значения от программы

Значение переменной может быть изменено в результате воздействия программы. Название (имя) переменной остаётся неизменным всегда.

2. ВИДЫ И ТИПЫ ПЕРЕМЕННЫХ

2.1 Типы данных C++

В языке C++ все переменные имеют определенный тип данных. Например, переменная, имеющая целочисленный тип не может содержать ничего кроме целых чисел, а переменная с плавающей точкой — только дробные числа.

Тип данных присваивается переменной при ее объявлении или инициализации.

Таблица 2.1

Типы данных C++

Таблица 1 — Типы данных C++

Тип	байт	Диапазон принимаемых значений
целочисленный (логический) тип данных		
bool	1	0 / 255
целочисленный (символьный) тип данных		
char	1	0 / 255
целочисленные типы данных		
short int	2	-32 768 / 32 767
unsigned short int	2	0 / 65 535
int	4	-2 147 483 648 / 2 147 483 647
unsigned int	4	0 / 4 294 967 295
long int	4	-2 147 483 648 / 2 147 483 647
unsigned long int	4	0 / 4 294 967 295

Таблица 1 — Типы данных C++

Тип	байт	Диапазон принимаемых значений
типы данных с плавающей точкой		
float	4	-2 147 483 648.0 / 2 147 483 647.0
long float	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0
double	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0

В таблице 2.1 представлены основные типы данных в C++. Вся таблица делится на три столбца. В первом столбце указывается зарезервированное слово, которое будет определять, каждое свой, тип данных. Во втором столбце указывается количество байт, которое отводится под переменную с соответствующим типом данных. В третьем столбце показан диапазон допустимых значений. В таблице все типы данных расположены от меньшего к большему.

Тип данных bool

Первый в таблице — это тип данных bool — целочисленный тип данных, так как диапазон допустимых значений — целые числа от 0 до 255. Но как Вы уже заметили, в круглых скобочках написано — логический тип данных, и это тоже верно. Так как bool используется исключительно для хранения результатов логических выражений. У логического выражения может быть один из двух результатов true или false. true — если логическое выражение истинно, false — если логическое выражение ложно[4].

Но так как диапазон допустимых значений типа данных bool от 0 до 255, то необходимо было как-то сопоставить данный диапазон с определёнными в языке программирования логическими константами true и false. Таким образом, константе true эквивалентны все числа от 1 до 255 включительно, тогда как

константе false эквивалентно только одно целое число — 0. Рассмотрим программу с использованием типа данных bool.

```
1  // data_type.cpp: определяет точку входа для консольного приложения.
2
   #include "stdafx.h"
3
   #include <iostream>
4
   using namespace std;
5
6  int main(int argc, char* argv[])
7  {
8      bool boolean = 25; // переменная типа bool с именем boolean
9      if ( boolean ) // условие оператора if
10         cout << "true = "  << boolean << endl; // выполнится в случае истинности
   условия
11     else
12         cout << "false = " << boolean << endl; // выполнится в случае, если условие
13 ложно
14     system("pause");
15     return 0;
16 }
```

В строке 9 объявлена переменная типа bool, которая инициализирована значением 25. Теоретически после строки 9, в переменной boolean должно содержаться число 25, но на самом деле в этой переменной содержится число 1. Число 0 — это ложное значение, число 1 — это истинное значение. Суть в том, что в переменной типа bool могут содержаться два значения — 0 (ложь) или 1 (истина). Тогда как под тип данных bool отводится целый байт, а это значит, что переменная типа bool может содержать числа от 0 до 255. Для определения ложного и истинного значений

необходимо всего два значения 0 и 1. Возникает вопрос: «Для чего остальные 253 значения?».

Исходя из этой ситуации, договорились использовать числа от 2 до 255 как эквивалент числу 1, то есть истина. Вот именно по этому в переменной boolean содержится число 25 а не 1. В строках 10 -13 объявлен оператор условного выбора if, который передает управление оператору в строке 11, если условие истинно, и оператору в строке 13, если условие ложно. Результат работы программы смотреть на рисунке 2.1

Рис. 2.1.Тип данных bool

Тип данных char

Тип данных char — это целочисленный тип данных, который используется для представления символов. То есть, каждому символу соответствует определённое число из диапазона [0;255]. Тип данных char также ещё называют символьным типом данных, так как графическое представление символов в C++ возможно благодаря char. Для представления символов в C++ типу данных char отводится один байт, в одном байте — 8 бит, тогда возведем двойку в степень 8 и получим значение 256 — количество символов, которое можно закодировать. Таким образом, используя тип данных char можно отобразить любой из 256 символов. Все закодированные символы представлены в таблице ASCII[5].

ASCII (от англ. American Standard Code for Information Interchange) — американский стандартный код для обмена информацией.

Рассмотрим программу с использованием типа данных char.

```
1  // symbols.cpp: определяет точку входа для консольного приложения.
2  #include "stdafx.h"
3  #include <iostream>
4  using namespace std;
5
6  int main(int argc, char* argv[])
7  {
8      char symbol = 'a'; // объявление переменной типа char и инициализация её символом 'a'
9      cout << "symbol = " << symbol << endl; // печать символа, содержащегося в
10 переменной symbol
11     char string[] = "cppstudio.com"; // объявление символьного массива (строки)
12     cout << "string = " << string << endl; // печать строки
13     system("pause");
14     return 0;
15 }
```

Итак, в строке 9 объявлена переменная с именем symbol, ей присвоено значение символа 'a' (ASCII код). В строке 10 оператор cout печатает символ, содержащийся в переменной symbol. В строке 11 объявлен строковый массив с именем string, причём размер массива задан неявно. В строковый массив сохранена строка "cppstudio.com". Обратите внимание на то, что, когда мы сохраняли символ в переменную типа char, то после знака равно мы ставили одинарные кавычки, в которых и записывали символ. При инициализации строкового массива некоторой строкой, после знака равно ставятся двойные кавычки, в которых и записывается некоторая строка. Как и обычный символ, строки выводятся с помощью оператора cout, строка 12. Результат работы программы показан на рисунке 2.2

Рис. 2 — Тип данных char

Целочисленные типы данных

Целочисленные типы данных используются для представления чисел. В таблице 1 их аж шесть штук: short int, unsigned short int, int, unsigned int, long int, unsigned long int. Все они имеют свой собственный размер занимаемой памяти и диапазоном принимаемых значений. В зависимости от компилятора, размер занимаемой памяти и диапазон принимаемых значений могут изменяться. В таблице 2.1 все диапазоны принимаемых значений и размеры занимаемой памяти взяты для компилятора MVS2010. Причём все типы данных в таблице 1 расположены в порядке возрастания размера занимаемой памяти и диапазона принимаемых значений. Диапазон принимаемых значений, так или иначе, зависит от размера занимаемой памяти. Соответственно, чем больше размер занимаемой памяти, тем больше диапазон принимаемых значений. Также диапазон принимаемых значений меняется в случае, если тип данных объявляется с приставкой unsigned — без знака. Приставка unsigned говорит о том, что тип данных не может хранить знаковые значения, тогда и диапазон положительных значений увеличивается в два раза, например, типы данных short int и unsigned short int[6].

Приставки целочисленных типов данных:

- short — приставка укорачивает тип данных, к которому применяется, путём уменьшения размера занимаемой памяти;
- long — приставка удлиняет тип данных, к которому применяется, путём увеличения размера занимаемой памяти;
- unsigned (без знака) — приставка увеличивает диапазон положительных значений в два раза, при этом диапазон отрицательных значений в таком типе данных храниться не может.

Так, что, по сути, имеем один целочисленный тип для представления целых чисел — это тип данных int. Благодаря приставкам short, long, unsigned появляется некоторое разнообразие типов данных int, различающихся размером занимаемой памяти и (или) диапазоном принимаемых значений.

Типы данных с плавающей точкой

В C++ существуют два типа данных с плавающей точкой: float и double. Типы данных с плавающей точкой предназначены для хранения чисел с плавающей точкой. Типы данных float и double могут хранить как положительные, так и отрицательные числа с плавающей точкой. У типа данных float размер занимаемой памяти в два раза меньше, чем у типа данных double, а значит и диапазон принимаемых значений тоже меньше. Если тип данных float объявить с приставкой long, то диапазон принимаемых значений станет равен диапазону принимаемых значений типа данных double. В основном, типы данных с плавающей точкой нужны для решения задач с высокой точностью вычислений, например, операции с деньгами.

Итак, были рассмотрены главные моменты, касающиеся основных типов данных в C++. Осталось только показать, откуда взялись все эти диапазоны принимаемых значений и размеры занимаемой памяти. А для этого разработаем программу, которая будет вычислять основные характеристики всех, выше рассмотренных, типов данных.

```
// data_types.cpp: определяет точку входа для консольного приложения.

#include "stdafx.h"

#include <iostream>

// библиотека манипулирования вводом/выводом

#include <iomanip>

// заголовочный файл математических функций
1
#include <cmath>
2
using namespace std;
3

4 int main(int argc, char* argv[])

5 {

6     cout << "     data type      "   << "byte"                    << "      "    << "    max
value  " << endl // заголовки столбцов
7
            << "bool              =  " << sizeof(bool)              << "        " << fixed <<
8 setprecision(2)

9 /*вычисляем максимальное значение для типа данных bool*/                      <<
10 (pow(2,sizeof(bool) * 8.0) - 1)             << endl

11         << "char              =  " << sizeof(char)              << "        " << fixed <<
setprecision(2)
12
/*вычисляем максимальное значение для типа данных char*/                      <<
13 (pow(2,sizeof(char) * 8.0) - 1)             << endl

14         << "short int         =  " << sizeof(short int)         << "        " << fixed <<
15 setprecision(2)

16 /*вычисляем максимальное значение для типа данных short int*/                  <<
(pow(2,sizeof(short int) * 8.0 - 1) - 1)      << endl

17
            << "unsigned short int =  " << sizeof(unsigned short int)  << "        " <<
18 fixed << setprecision(2)

19 /*вычисляем максимальное значение для типа данных unsigned short int*/
```

Оператор sizeof() вычисляет количество байт, отводимое под тип данных или переменную. Функция pow(x,y) возводит значение x в степень y, данная функция доступна из заголовочного файла <cmath>. Манипуляторы fixed и setprecision() доступны из заголовочного файла <iomanip>. Первый манипулятор — fixed, передаёт в поток вывода значения в фиксированной форме.
Манипулятор setprecision(n) отображает n знаков после запятой. Максимальное значение некоторого типа данных вычисляется по такой формуле:

```
max_val_type = 2^(b * 8 - 1) - 1; // для типов данных с отрицательными и положительными числами

// где, b - количество байт выделяемое в памяти под переменную с таким типом данных

// умножаем на 8, так как в одном байте 8 бит

// вычитаем 1 в скобочках, так как диапазон чисел надо разделить надвое для положительных и отрицательных значений

// вычитаем 1 в конце, так как диапазон чисел начинается с нуля

// типы данных с приставкой unsigned

max_val_type = 2^(b * 8 ) - 1; // для типов данных только с положительными числами

// пояснения к формуле аналогичные, только в скобочка не вычитается единица
```

Пример работы программы можно увидеть на рисунке 2.3.

```
    data type      byte          max value
bool               =  1         255.00
char               =  1         255.00
short int          =  2         32767.00
unsigned short int =  2         65535.00
int                =  4         2147483647.00
unsigned int       =  4         4294967295.00
long int           =  4         2147483647.00
unsigned long int  =  4         4294967295.00
float              =  4         2147483647.00
double             =  8         9223372036854775808.00
Для продолжения нажмите любую клавишу . . .
```

Рис.2. 3 — Типы данных C++

В первом столбце показаны основные типы данных в C++, во втором столбце размер памяти, отводимый под каждый тип данных и в третьем столбце — максимальное значение, которое может содержать соответствующий тип данных. Минимальное значение находится аналогично максимальному[7]. В типах данных с приставкой unsigned минимальное значение равно 0.

```
1 // сокращённая запись типа данных int
2 short a1; //  тоже самое, что и short int
3 long a1; // тоже самое, что и long int
4 unsigned a1; // тоже самое, что и unsigned int
5 unsigned short a1; // тоже самое, что и unsigned short int
```

Если, например, переменной типа short int присвоить значение 33000, то произойдет переполнение разрядной сетки, так как максимальное значение в переменной типа short int это 32767. То есть в переменной типа short int сохранится какое-то другое значение, скорее всего будет отрицательным. Так как затронули тип данных int, стоит отметить, что можно опускать ключевое слово int и писать, например, просто short. Компилятор будет интерпретировать такую запись как short int. Тоже самое относится и к приставкам long и unsigned.

2.2 Объявление переменных в C++

Чтобы использовать переменную в C++, необходимо сначала объявить это, указывая, каким типом данных мы хотим, чтобы это было. Синтаксис, чтобы объявить новую переменную должен записать спецификатор требуемого типа данных (как интервал, bool, плавание...) сопровождаемый допустимым идентификатором переменной. Например:

```
1 int a;
2 float mynumber;
```

Они - два допустимых объявления переменных. Первый объявляет переменную типа int с идентификатором a. Второй объявляет переменную типа float с идентификатором mynumber. После того, как объявленный, переменные a и mynumber могут использоваться в пределах остальной части их контекста в программе.

Если нужно объявить больше чем одну переменную того же самого типа, можно объявить всех их в единственном операторе, разделяя их идентификаторы цезурами. Например:

```
int a, b, c;
```

Это объявляет три переменные (a, b и c), все они типа int, и имеет точно то же самое значение как:

```
int a;
int b;
int c;
```

Целочисленные типы данных char, short, long и int могут быть или подписаны или без знака в зависимости от диапазона чисел, должен был быть представлен. Типы со знаком могут представить и положительные и отрицательные величины, тогда как типы без знака могут только представить положительные значения (и нуль). Это может быть указано или при использовании спецификатора signed или при использовании спецификатора unsigned перед именем типа. Например:

```
unsigned short int NumberOfSisters;
signed int MyAccountBalance;
```

По умолчанию, если не указываем или signed или unsigned, большинство настроек компилятора предположит, что тип подписывается, поэтому вместо второго объявления выше, мы, возможно, записали:

```
int MyAccountBalance;
```

с точно тем же самым значением (с или без ключевого слова signed). Исключение к этому общему правилу - тип char, который существует отдельно и считается различным фундаментальным типом данных от signed char и unsigned char, который, как думают, сохранил символы. Следует использовать также signed или unsigned если нужно сохранить численные значения в char - доведенная до требуемого размера переменная.

short и long может использоваться один в качестве спецификаторов типа. В этом случае они обращаются к их соответствующим целочисленным фундаментальным типам: short эквивалентно short int и long эквивалентно long int. Следующие два объявления переменной эквивалентны:

```
1 short Year;
2 short int Year;
```

Наконец, signed и unsigned май также использоваться в качестве автономных спецификаторов типа, означая то же самое как signed int и unsigned int соответственно. Следующие два объявления эквивалентны:

```
1 unsigned NextYear;
2 unsigned int NextYear;
```

Контекст переменных

Все переменные, которые мы намереваемся использовать в программе, должно быть, были объявлены с ее спецификатором типа в более ранней точке в коде, как было сделано в предыдущем коде в начале корпуса функционального основного, когда мы объявили, что a, b, и result имели тип int.

Переменная может быть или глобального или локального контекста. Глобальная переменная - переменная, объявленная в основном корпус исходного кода вне всех функций, в то время как локальная переменная - та, объявленная в пределах

корпуса функции или блока[8].

Глобальные переменные могут быть отнесены отовсюду в коде, четных внутренних функциях, всякий раз, когда это после его объявления.

Контекст локальных переменных ограничивается блоком, включенным в фигурные скобки ({}) где они объявляются. Например, если они объявляются в начале корпуса функции (как в функциональном main), их контекст между его местом объявления и концом той функции. В примере выше, это означает, что, если другая функция существовала в дополнение к main, к локальным переменным, объявленным в main, нельзя было бы получить доступ от другой функции и наоборот.

Инициализация переменных

Объявляя регулярную локальную переменную, ее значение является по умолчанию неопределенным. Но можно хотеть, чтобы переменная сохранила конкретное значение одновременно, что это объявляется. Чтобы сделать это, можно инициализировать переменную. Есть два способа сделать это в C++:

Первый, известный как подобная до инициализация, делается, добавляя знак "равно", сопровождаемый значением, к которому будет инициализирована переменная:

type identifier = initial_value ;

Например, если нужно объявить переменную int под названием a, инициализированный со значением 0 сейчас, в котором он объявляется, то могли записать:

```
 int a = 0;
```

Другой способ инициализировать переменные, известные как инициализация строителя, делается, включая первоначальное значение между круглыми скобками (()):

type identifier (initial_value) ;

Например:

```
int a (0);
```

Оба способа инициализировать переменные допустимы и эквивалентны в C++.

```
// initialization of variables
#include <iostream>
using namespace std;
int main ()
{
int a=5; // initial value = 5
int b(2); // initial value = 2                    6
int result; // initial value undetermined
a = a + 3;
result = a - b;
cout << result;
return 0;
}
```

Вводная часть к строкам

Переменные, которые могут сохранить не численные значения, которые более длинны чем один одиночный буквенно-цифровой символ, известны как строки.

Библиотека языка C++ оказывает поддержку для строк через норму string класс. Это не фундаментальный тип, но он ведет себя похожим способом, как фундаментальные типы делают в его самом основном использовании[9].

Первое различие для фундаментальных типов данных то, что, чтобы объявить и использовать объекты (переменные) этого типа, должны включать

дополнительный заголовочный файл в наш исходный код: <string> и иметь доступ к std пространство имен (который уже имели во всех наших предыдущих программах благодаря using namespace оператор).

```
// my first string

#include <iostream>

#include <string>

using namespace std;

int main ()
                                        This is a string
{

string mystring = "This is a string";

cout << mystring;

return 0;

}
```

Поскольку можно видеть в предыдущем примере, строки могут быть инициализированы с любым допустимым строковым литералом точно так же, как численные переменные типа могут быть инициализированы к любому допустимому численному литералу. Оба формата инициализации допустимы со строками:

```
string mystring = "This is a string";

string mystring ("This is a string");
```

Строки могут также выполнить все другие основные операции, что фундаментальные типы данных могут, как то, чтобы быть объявленным без первоначального значения и присвоенных значений во время выполнения:

```
// my first string
#include <iostream>
#include <string>
using namespace std;
int main ()
{
string mystring;
mystring = "This is the initial string content";
cout << mystring << endl;
mystring = "This is a different string content";
cout << mystring << endl;
return 0;
}
```

```
This is the initial string content
This is a different string content
```

ЗАКЛЮЧЕНИЕ

Таким образом, в результате проведенного исследования можно сделать следующие выводы

Под переменной в языках программирования понимают программный объект (число, слово, часть слова, несколько слов, символы), имеющий имя и значение, которое может быть получено и изменено программой.

В программировании переменная (variable) это своего рода емкость для хранения данных. Когда информация записана в переменной (или по-другому, когда переменной присвоено значение), тогда эту информацию можно изменять, выводить в окне Web-броузера, посылать по электронной почте и т.д.

Свойством переменной является её способность получать некоторое значение от программы, удерживать его в течение времени работы программы и сообщать это значение программе при запросах программы.

Существует множество различных типов переменных, почти каждый язык программирования имеет свои различные типы переменных.

Основные типы данных в C++

- int — целочисленный тип данных.
- float — тип данных с плавающей запятой.
- double — тип данных с плавающей запятой двойной точности.
- char — символьный тип данных.
- bool — логический тип данных.

Пример объявления переменных

- int a; // объявление переменной a целого типа.
- float b; // объявление переменной b типа данных с плавающей запятой.
- double c = 14.2; // инициализация переменной типа double.
- char d = 's'; // инициализация переменной типа char.
- bool k = true; // инициализация логической переменной k.

В C++ оператор присваивания (=) — не является знаком равенства и не может использоваться для сравнения значений. Оператор равенства записывается как «двойное равно» — ==.

Присваивание используется для сохранения определенного значение в переменной. Например, запись вида a = 10 задает переменной a значение числа 10.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Безручко В.Т., Информатика курс лекций / В.Т. Безручко.–М.: ИНФРА–М. 2013–432.с.
2. Волков В. Б., Макарова Н. В. Информатика для ВУЗов – СПб.: «Питер», 2014
3. Информатика: Учебник /Под редакцией В.М. Мартю. – 2014г.
4. Комлев Н.Ю. Объектно Ориентированное Программирование – М.: «Дашков и К», 2014

5. Левин М.П. Параллельное программирование с использованием OpenMP – М.: «Интернет-университет информационных технологий, Бином. Лаборатория знаний», 2012
6. Литвиненко Н. А. Технология программирования на C++ – СПб. : «БХВ-Петербург», 2013
7. Медведев В. И. Особенности объектно-ориентированного программирования на C++/CLI, C# и Java – М.: Издательство: РИЦ «Школа», 2010
8. Орлов С.А. Теория и практика языков программирования – СПб.: «Питер», 2014
9. Прата С. Язык программирования C++. Лекции и упражнения, 6-е издание – Издательство: Вильямс, 2012
10. Рубальская О.Н. Информатика. Самоучитель на CD: Учеб. Пособие.–М.: ИД. "Форум": ИНФРА– М. 224с.
11. Титов В.М., Информатика для экономистов: учебник /В.М. Титов.– М.: ИНФРА–М. 2013–448с.
12. Хортон А. Visual C++ – М.: «Издательство: Вильямс», 2011

1. Информатика: Учебник /Под редакцией В.М. Мартю. – 2014г. ↑

2. Левин М.П. Параллельное программирование с использованием OpenMP – М.: «Интернет-университет информационных технологий, Бином. Лаборатория знаний», 2012 ↑

3. Титов В.М., Информатика для экономистов: учебник /В.М. Титов.– М.: ИНФРА–М. 2013–448с. ↑

4. Комлев Н.Ю. Объектно Ориентированное Программирование – М.: «Дашков и К», 2014 ↑

5. Волков В. Б., Макарова Н. В. Информатика для ВУЗов – СПб.: «Питер», 2014 ↑

6. Хортон А. Visual C++ – М.: «Издательство: Вильямс», 2011 ↑

7. Прата С. Язык программирования C++. Лекции и упражнения, 6-е издание – Издательство: Вильямс, 2012 ↑

8. Орлов С.А. Теория и практика языков программирования – СПб.: «Питер», 2014 ↑

9. Литвиненко Н. А. Технология программирования на C++ – СПб. : «БХВ-Петербург», 2013 ↑