Міністерство освіти і науки України
Вищий приватний навчальний заклад
Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука

А.В Ільчук

Принципи типізації даних

Науковий керівник:
Р.М.Літнарович, доцент,к.т.н.

Рівне – 2012 р.

УДК 004.43(076)
Ільчук А.В. Принципи типизації даних. МЕГУ, Рівне, 2012р-60с. Ilchuk A.V. Principles of tipizacii of information. IEGU, Rivne, 2012p-60 p.
Рецензенти: В.Г.Бурачек, доктор технічних наук, професор
Є.С. Парняков, доктор технічних наук, професор
В.О.Боровий , доктор технічних наук, професор
Відповідальний за випуск: Й.В. Джунь, доктор фізико-математичних наук, професор.
Послідовно розглядаються основні поняття побудови сучасного програмування. Монографія містить актуальний матеріал довідково-аналітичного характеру по наступних темах: прості типии даних, прості змінні, цілочисельні типии даних., типи даних short, int і long, програма limits.cpp, типи даних без знака , програма exceed.cpp цілочисельні константи, програма hexoct.cpp, тип даних char: символи і малі цілі числа, програма chartype.cpp, програма morechar.cpp, функція-елемент: cout.put, константи типу char, програма bondini.cpp, типи даних signed char і unsigned char, типи даних wchar_t4, тип даних boo, типи даних з плаваючою крапкою, програма floatnum.cpp, константи з плаваючою крапкою, переваги і недоліки типів з плаваючою
Ключові слова: Прості типии даних , цілочисельні типии даних, тип даних char: символи і малі цілі числа, тип даних bool, типи даних з плаваючою крапкою, теретворення типів даних.

Последовательно рассматриваются основные понятия построения современного программирования. Монография содержит актуальный материал справочно аналитического характеру по следующим темам: простые типии данных, простые переменные, целочисленные типии данных., типы

данных short, int и long, программа limits.cpp, типы данных без знака, программа exceed.cpp целочисленные константы, программа hexoct.cpp, тип данных char: символы и малые целые числа, программа chartype.cpp, программа morechar.cpp, функция-элемент: cout.put, константы типа char, программа bondini.cpp, типы данных signed char и unsigned char, типы данных wchar_t4, тип данных boo, типи данных, с плавающей точкой, программа floatnum.cpp, константы с плавающей точкой, преимущества и недостатки типов с плавающей

Ключевые слова: Простые типии данных, целочисленные типии данных, тип данных char: символы и малые целые числа, тип данных bool, типи данных, с плавающей точкой, теретворення типов данных.

The basic concepts of construction of the modern programming are consistently examined. A monograph contains actual material certificate analytical to character on the followings themes: simple types of information, scalar variables, fullnumber types of information., types of data of short, int and long, program of limits.cpp, types of data without a sign, program of exceed.cpp fullnumber constants, program of hexoct.cpp, type of data of char: characters and small integers, program of chartype.cpp, program of morechar.cpp, function-element: cout.put, constants of type of char, program of bondini.cpp, types of data of signed char and unsigned char, types of data of wchar_t4, type of information of boo, types of information, with a floating point, program of floatnum.cpp, constants with a floating point, advantages and lacks of types with floating.

Keywords: Simple types of information, fullnumber types of information, type of data of char: characters and small integers, tip of information of bool, types of information, with a floating point, transformation types of data.

Ільчук Андрій Васильович

Спеціаліст системотехнік, магістрант
інформаційних технологій.

Ільчук Андрій Васильович

Спеціаліст системотехнік, магістрант інформаційних технологій
ІН-11М

Принципи типізації даних

Комп'ютерний набір, верстка і макетування та дизайн в редакторі Microsoft®Office® Word 2003 А.В.Ільчук.

Науковий керівник Р. М. Літнарович, доцент, кандидат технічних наук

Міжнародний Економіко-Гуманітарний Університет ім. акад. Степана Дем'янчука

Кафедра математичного моделювання

33027,м.Рівне,Україна

Вул.акад. С.Дем'янчука,4, корпус 1

Телефон:(+00380) 362 23-73-09

Факс:(+00380) 362 23-01-86

E-mail:mail@regi.rovno.ua

E-mail:ururu_88@mail.ru

Зміст

Вступ.....7

1. Прості типии даних……10

1.1. Прості змінні……12

2. Цілочисельні типии даних……..14

2.1. Типи даних short, int і long……..15

2.2. додаток 1. Програма limits.cpp……18

2.3. Типи даних без знака……22

2.4. додаток 3. Програма exceed.cpp……...24

2.5. Цілочисельні константи……26

2.6. додаток 3. Програма hexoct.cpp……27

3. Тип даних char: символи і малі цілі числа……29

3.1. додаток 4. Програма chartype.cpp……30

3.2. додаток 5. Програма morechar.cpp……...32

3.3. Функція-елемент: cout.put ()……34

3.4. Константи типу char……..36

3.5. додаток 6. Програма bondini.cpp……39

3.6. Типи даних signed char і unsigned char……40

3.7. Ти даних wchar_t……...41

4. Тип даних bool……..43

5. Типи даних з плаваючою крапкою……45

5.1. додаток 7. Програма floatnum.cpp……47

5.2. Константи з плаваючою крапкою……49

5.3. Переваги і недоліки типів з плаваючою крапкою……50

5.4. додаток 8. Програма fltadd.cpp…….50
6. Перетворення типів даних……..…52
6.1. Перетворення при привласненні…….53
6.2. додаток 9. Програма assign.cpp……...55
Висновки.....57
Літературні джерела:……...59

Літературні джерела

1. Основи сучасного програмування. О. Бичков, *Ю. Турбал.*
2. C/C++ в задачах и примерах. Н. Культин.
3. Язык программирования C++ Стивен Прата
4. Грис Д. Наука программирования.- М., 1984.
5. Хортон А., Visual C++ 2005: базовый курс. – С.Пб.: Диалектика, 2007. -1152 ст.
6. Грегори К., Использование Visual C++ 6. Специальное издание. –М.: Вильямс, 2000. - 864ст.
7. Сергеев А.П., Терен А.Н., Программирование в Microsoft Visual C++ 2005. Самоучитель. –С.Пб.: Диалектика, 2006. -352 ст.
8. Солтер Н., Клепер Дж., C++ для профессионалов . –С.Пб.: Диалектика, 2006. -912 ст.
9. Шилдт Г. , C++: базовый курс, 3-е издание . –М.: Вильямс, 2005. -624 ст.
10. Шилдт Г. , Справочник программиста по C/C++, 3-е издание. –М.: Вильямс, 2003. -432 ст.
11. Э. Ишкова «C++ начала программирования» - М.: Бином, 2001.
12. Романов Е. Л. «Практикум по программированию на C++: Уч. Пособие» 2004р – 433с
13. Бублик В.В. Методические указания и задания к лабораторным занятиям по курсу "Вычислительные машины и программирование".- Киев, 1986
14. Ільчук А.В. Технології комп’ютерної безпеки.

    Книга 4. МЕГУ, Рівне, 2011.-73 с.

визначає, яким чином трактується код: як символ або як число.

Типи даних з плаваючою крапкою служать для представлення дробових чисел і забезпечують представлення набагато більших чисел, ніж типи цілочисельних даних. Існує три типи даних з плаваючою крапкою: float, double і long double. В мові С++ величина даних типу float повинна бути не більше, ніж розміри даних типу double, а величина даних типу double — не більше, ніж величина даних типу long double. Як правило, дані типу float займають 32 біти пам'яті, дані типу double – 64 біта і дані типу long — від 80 до 128 бітів. Наявність в мові С++ великої кількості типів даних різних величин, а також із знаком і без знака дає можливість програмісту вибрати той тип даних, який краще всього відповідає конкретним вимогам, що пред'являються до даних.

Можливо, велика кількість основних типів даних в мові С++ спочатку покажеться надмірною, особливо якщо взяти до уваги різні правила перетворення даних. Проте найімовірніше, рано, або пізно це одінеться, за ситуації, коли один з багатьох типів даних виявиться самим відповідним в даний момент.

ВСТУП

Застосування програм у самих різних областях людської діяльності привело до необхідності підвищення надійності всього програмного забезпечення. Одним з напрямків підвищення надійності і, відповідно, вдосконалення мов програмування став розвиток типізації даних. Теорія типів даних виходить з того, що кожне з даних, які використовуються в програмі, належить одному і тільки одному типу. Тип даного визначає множину його можливих значень і набір операцій, що можуть виконуватись над цим даним. Дані конкретного типу в ряді випадків можуть бути перетворені в дані іншого типу. Такі перетворення можуть бути явними чи здійснюватись в ряді мов неявно.

Надійність програмного забезпечення є мірою ступеня автоматичного виявлення помилок, яке може бути виконаним або компілятором, або системою, що підтримує виконання скомпільованої програми.

Розглянемо один такий приклад. Припустимо, що масив a, що складається з 20 цілих чисел, індексується цілою змінною *i* таким чином, що *i-та* компонента масиву *a* позначається a[i]. Очевидно, що надійна програма повинна забезпечити виконання умови $1 < i < 20$ у будь-якому місці програми, де зустрічається a[i].

Для цього є дві можливості. Традиційний підхід полягає в простому внесенні в готову програму перед кожним звертанням до масиву додаткового коду, що перевіряє виконання необхідної умови. Більш сучасний підхід полягає в перевірці значення *i* не під час його використання, а скоріше під час присвоювання значення змінній. Це вимагає додаткових зусиль програміста, що повинен специфікувати область значень і при описі змінної. Знаючи, що заданий тип змінної і обмежує можливі значення областю від 1 до 20, немає необхідності взагалі перевіряти значення індексу при звертанні до a[i]. Замість цього перевіряються значення, що присвоюються змінній i. Дуже часто законність таких присвоювань може бути перевірена під час компіляції, так що загальний обсяг фактично необхідного об'єктного коду перевірки допустимих значень і істотно зменшується. Ще однією перевагою такого підходу є те, що при цьому зростає ясність програм, адже області, значень, які набуває кожна змінна, встановлюється явно.

Всі універсальні мови програмування, незважаючи на розходження в синтаксисі і ключових словах, реалізують ті самі канонічні структури: оператори присвоювання, цикли і розгалуження. В усіх сучасних мовах присутні визначені (базові) типи даних (цілі та дійсні арифметичні типи, символьний і, можливо, рядковий тип), є можливість

Висновки

Основні типи даних мови C++ підрозділяються на дві групи. Одна група складається з типів даних, які зберігаються як цілі числа. Друга група включає типи даних, які зберігаються у форматі чисел з плаваючою крапкою. Різні типи цілочисельних даних відрізняються один від одного кількістю елементів пам'яті, для зберігання чисел, що використовуються, а також наявністю або відсутністю знака. Ось всі типи цілочисельних даних (починаючи з найменшим і закінчуючи найбільшим): bobl, char, signed char, unsigned char, short, unsigned short, int, unsigned int, long і unsigned long. Є також тип даних wchar_t, положення якого в ряду залежить від реалізації мови. В мові C++ дані типу char повинні мати достатньо велику величину для представлення основного набору символів системи, дані типу wchar_t — достатньо велику величину для представлення розширеного набору символів системи, розмір даних типу short — не менше 16 бітів, розмір даних типу int— не менше розміру даних типу short, розмір даних типу long — не менше 32 бітів і не менше розміру даних типу int. Точна величина залежить, від реалізації мови.

Символи в комп'ютерах, представлені за допомогою своїх числових кодів. Система уведення-виведення

guess = 3

debt = 0

Тут змінній tree привласнюється значення з плаваючою крапкою 3.0. Проте оскільки об'єкт **cout** при висновку даних опускає завершальні нулі, то замість 3.0 відображається 3. При привласненні числа 3.9832 зміною guess типу **int** воно урізується до числа 3; в мові C++ при перетворенні даних з плаваючою крапкою в цілочисельні дані здійснюється урізування числа (відкидання дробової частини), а не округлення (заміна найближчим цілим числом). І нарешті, зверніть увага, що змінною debt типу int не може бути привласнено значення: 3,0E12. Результат такої ситуації в мові C++ не визначений! В нашій системі змінній debt привласнюється 0.

Деякі компілятори видають попередження, про можливу втрату даних в тих операторах, де цілочисельні змінні ініціалізуються числами з плаваючою крапкою. Крім того, значення змінної debts, що відображається, міняється від компілятора до компілятора. Наприклад, виконання цієї ж самої програми в іншій системі, дає для змінної **debts** значення 2112827392.

використання об'єктів даних, у тому числі масивів і структур (записів). Для арифметичних даних дозволені звичайні арифметичні операції, для об'єктів даних звичайно передбачена тільки операція присвоювання і можливість звертання до елементів.

1. Прості типи даних.

Вигляд інформації, представлений кожним словом даних у пам'яті, може бути різний. Наприклад, деякі слова можуть містити цілі числа, у той час як інші можуть містити символи. Сучасна мова високого рівня дає програмісту абстрактну модель, у якій дані й операції над даними можна специфікувати в об'єктно-орієнтованих поняттях.

При цьому мова високого рівня забезпечує наступні можливості доступу до об'єктів даних у пам'яті:

- на об'єкти даних посилаються за допомогою визначених користувачем імен (ідентифікаторів);
- об'єкти даних зв'язані з типом, що визначає множину значень, які можуть приймати об'єкти цього типу, і множину операцій, що можуть застосовуватися до об'єктів цього типу.

У ранніх мовах введення поняття типу даних було чисто прагматичним прийомом, який застосовували з метою полегшити роботу компілятору. І насправді користувач звертав мало уваги на поняття типу. Для сучасних мов розроблені дуже складні механізми типів, що дозволяють програмісту виразити розв'язок задачі з більшою ясністю і зі значно більш високою надійністю. Гарний механізм типів насправді є ключовим чинником при забезпеченні надійності мови програмування, а, як вказувалося раніше,

результат в даному випадку; це означає, що в різних реалізаціях результати можуть бути різними. В програмі з додатку 9. демонструється декілька перетворень при привласненні значень.

6.2. додаток 9. Програма assign.cpp

```
//Програма assign.cpp — зміна типу даних при привласненні
#include<iostream.h>
unsing namespace std;
int main()
{
float tree = 3;  //дані типу int перетворяться в дані типу float
int guess = 3.9832; //дані типу float перетворяться в дані типу
int
int debt = 3,0E12;  //в мові C++ результат не визначений
cout << “tree = ” << tree << “\n”;
cout << “guess = ” << guess << “\n”;
cout << “debt = ” << debt << “\n”;
return 0;
}
```

В результаті виконання програми в одній з систем виходить наступний результат:

tree = 3

розширює його до типу даних long (як правило, 32-бітове число). Зверніть увагу, що при розширенні створюється нове число, яке і привласнюється змінній so_long; вміст змінної thirty не змінюється.

Привласнення значення змінної, що має велику величину, зазвичай не викликає проблем. Наприклад, при привласненні значення типу short змінної типу long привласнюване значення не змінюється; воно просто доповнюється вільними бітами. Проте при привласненні великого значення типу long, наприклад 2.111222333, змінній типу float призводить до деякої втрати точності. Оскільки змінна типу float може мати тільки шість значущих цифр, привласнюване значення може бути закруглено до 2.11122E9.

Нульове значення, привласнюване змінній типу bool, перетвориться в значення false, а ненульове значення перетвориться в значення true.

При привласненні значень з плаваючою крапкою цілочисельним змінним виникає дві проблеми. По-перше, перетворення даних з плаваючою крапкою в цілочисельні дані приводить до урізування числа (відкиданню дробової частини). По-друге, значення типу float може бути дуже великим, щоб помістится в змінній типу int, що має меншу величину. В мові C++ не визначено, яким повинен бути

необхідність надійності має першорядну важливість при програмуванні в реальному часі.

Сучасні мови забезпечують адекватну множину типів даних і операцій для розв'язку задач в обраній області. У мові є множина визначених типів даних (прості типи) і операцій та визначені механізми для специфікації типів, визначених користувачем. Наступні чотири простих типи даних є загальними для більшості мов:

- цілий
- дійсний
- логічний
- символьний.

Розглянемо можливості, що забезпечуються мовами програмування для типізації даних.

- ✓ Кожен тип визначає множину значень і множену операцій над даними цього типу.
- ✓ У кожній операції присвоювання тип значення, що присвоюється і тип об’єкта даних , якому здійснюється присвоювання, повинні бути еквівалентні.
- ✓ Кожна операція, що застосовується до об’єкта даних, повинна належати множені операцій, обумовленій типом об’єкта.

Наприклад, нехай ми маємо наступні описи змінних:

float x;

```
int i;
char c;
```

де float, int, char – ідентифікатори дійсного, цілого та символьного типів відповідно. Тоді присвоювання виду

```
i=x;
c=x;
```

не коректні, а

```
x=3.5;
i=7;
```

цілком законні з погляду синтаксису мови.

Природно, що повинен бути механізм перетворення типів для виконання, наприклад, таких присвоювань:

```
x=(float) i,
```

де ім’я типу float застосовується до цілого значення i та дає дійсний результат.

Виконання зазначених властивостей підвищує надійність і ясність програми. Всім операціям на абстрактному рівні мови забезпечується коректність.

1.1. Прості змінні

Як правило, програми повинні берегти яку-небудь інформацію. Для зберігання в комп'ютері елемента інформації програма повинна відстежувати три основні властивості цього, елемента інформації, зокрема, вона визначає:


```
char c;
int i;
c=I;
```

У випадку присвоювання більшого типу до меншого відбувається відкидання старших бітів (для цілих типів). У випадку присвоювання

меншого типу до більшого відбувається розширення копією знакового біта або нуля. У випадку присвоювання плаваючого типу до цілого

відбувається перетворення плаваючого типу до типу int, а потім

від типу int до типу операнда, що стоїть в лівій частині операції присвоювання.

6.1. Перетворення при привласненні

В мові C++ можна вільно числові дані одного типу привласнювати змінній іншого типу. При цьому привласнюванні дані перетворяться до типу змінної. Наприклад, припустимо, що є змінна so_long типа long і змінна thirty типу short, - а в програмі присутній наступний оператор:

```
so_long = thirty;   //привласнення типу данных short змінною типу long
```

При виконанні цього привласнення програма бере значення змінної thirty (як правило, 16-бітове число) і

6. Перетворення типів даних

Різноманітність типів даних в мові C++ дає програмісту можливість вибрати тип даних, відповідний конкретній потребі. Наприклад, складання двох чисел типу short може виконуватися за допомогою інших апаратних інструкцій, ніж складання двох чисел типу long. Коли є 11 типів цілочисельних даних і три типи даних з плаваючою крапкою, комп'ютеру доводиться обробляти безліч різних випадків, особливо, якщо в одній операції змішуються дані різних типів. Щоб не допустити можливої мішанини, в мові C++ багато перетворень типів даних виконуються автоматично:

- Перетворення даних здійснюється, коли дані одного арифметичного типу привласнюються змінній іншого арифметичного типу.
- Перетворення даних здійснюється, коли у виразі містяться дані різних типів.
- Перетворення даних здійснюється при передачі аргументів у функції.

Якщо операнди операцій належать до різних типів даних, то компілятор автоматично здійснює приведення типу до одного спільного. Операнди з меншим діапазоном, як правило приводяться до операндів з більшим діапазоном:

- де зберігається інформація
- яке значення там зберігається
- який вид інформації зберігається

Раніше в прикладах програм використовувався наступний спосіб оголошення змінної. Використаний в оголошенні тип даних описує вид інформації, а ім'я змінної є символічним представленням значення змінної. Припустимо, наприклад, що програміст використовує наступні оператори:

```
int braincount;
braincount = 5;
```

Ці оператори означають, що програма запам'ятовує ціле число і що ім'я braincount представляє це ціле число, в даному випадку 5. В єстві, програма виділяє область пам'яті: досить велику, щоб зберігати ціле число, відзначає її, привласнює цій області мітку braincount і копіює в неї значення 5. Ці оператори не повідомляють вас (або програмісту), де в пам'яті зберігається значення, але програмі відоме місцеположення цієї інформації. Дійсно, використовуючи операцію *&*, можна дізнатися адресу змінної braincount з пам'яті.

2. Цілочисельні типи даних

Цілі числа — це числа без дробової частини, наприклад, 2, 98, -5286, 0. В природі існує величезна кількість цілих чисел (ця кількість є нескінченною), тому ні в якій найбільшій оперативній пам'яті комп'ютера не можуть бути представлені всі можливі цілі числа. Отже, в будь-якій мові програмування може бути представлена тільки частина всіх цілих чисел. В деяких мовах програмування, наприклад в мові Standard Pascal, існує тільки один тип цілочисельних даних (один тип даних служить для представлення всіх цілих чисел!), але в мові C++ є декілька типів цілочисельних даних. Це дозволяє програмістам вибирати такий тип цілочисельних даних, який краще всього відповідає вимогам конкретної програми. Подібна турбота про те, щоб тип даних відповідав конкретним вимогам, полегшує створення нових типів даних в ООП.

В мові C++ різні типи цілочисельних даних розрізняються об'ємом пам'яті, що використовується для їх зберігання. Більший блок пам'яті може представляти більший діапазон цілих чисел. Крім того, одні типи даних (типи даних із знаком) можуть представляти і позитивні, і негативні значення, тоді як інші типи даних (типи даних без знака) не можуть представляти негативні значення. Основні


```
    cout  << “b - a =” << “b - a ” << “\n”;
    return  0;
}
```

В програмі до деякого числа додається 1, а потім з суми віднімається початкове число. В результаті повинна вийти одиниця. Чи не так? Ось результат, отриманий в одній з обчислювальних систем:

a = 2.34e+022

b - a = 0

Проблема полягає в тому, що 2.34E+22 — це число з 23 значущими цифрами зліва від десяткової крапки. Додаючи до нього одиницю, ми намагаємося додати одиницю до 23-й цифрі цього числа. Але тип даних float має тільки шість або сім значущих цифр, тому спроба змінити 23-у цифру не надає на число ніякого впливу.

5.3. Переваги і недоліки типів даних з плаваючою крапкою

У типів даних з плаваючою крапкою є дві переваги в порівнянні з цілочисельними даними. По-перше, дробові числа можуть бути представлений тільки типами даних з плаваючою крапкою. По-друге, у зв'язку з наявністю експоненти (степені), типи даних з плаваючою крапкою мають набагато більший діапазон значень. З другого боку, операції над типами даних з плаваючою крапкою, виконуються повільніше, ніж операції над цілочисельними данными, крім того, для типів даних з плаваючою крапкою може відбутися втрата точності. Останнє положення ілюструється в програмі fltadd.cpp з додатку 8.

5.4. додаток 8. Програма fltadd.cpp

```
// програма fltadd.cpp! — проблеми, зв'язані з втратою точності даних типу float
#include <iostream.h>
using namespace std;
int main()
{
        float   a = 2.34E+22f;
        float  b = a  +   l.Of;
        cout << “a=” << a << “\n”;
```


типи цілочисельних даних мови C++ (в порядку зростання розміру) іменуються **char, short, int і long.** Кожний з цих типів даних підрозділяється на два різновиди: із знаком і без знака. В результаті ви маєте на вибір вісім різних типів цілочисельних даних! Давайте розглянемо ці типи цілочисельних даних більш детально. Оскільки тип даних char володіє деякими особливими властивостями (частіше за все він використовується для представлення символів, а не чисел). Розглянемо його пізніше.

2.1. Типи даних short, int і long

Оперативна пам'ять комп'ютера складається з елементів, які називаються *бітами.* Маючи в своєму розпорядженні різне число бітів для зберігання значень, типи даних **short, int** і **long** можуть представляти цілі числа трьох різних величин. Було б зручно, як би дані кожного типу у всіх системах були однієї певної величини; наприклад, величина даних типу **short** завжди б була рівна 16 бітам, типу **int** - 32 бітам і тд. Але не все так просто. А, все тому що жоден варіант не підходить для всіх типів комп'ютерів. В мові C++ представляється гнучкий стандарт з деякими гарантованими мінімальними розмірами:

- Величина даних типу **short** не менше 16 бітів;

• Величина даних типу **int** не менше розміру даних типу short;

• Величина даних типу **long** не менше 32 бітів і не менше розміру даних типу **int.**

В даний час в багатьох системах використовуються мінімальні гарантовані розміри даних і дані типу short мають величину, рівну 16 бітам, а дані типу long – величину, рівну 32 бітам. Для даних типу int залишається можливість вибору. Їх величина може бути рівна 16, 24 або 32 бітам і при цьому відповідати стандарту. В більш ранніх реалізаціях мови C++ на комп'ютерах IBM PC дані типу int мають, як правило, величину 16 бітів (таку ж, як і у даних типу short); В Windows 95, Windows 98, Windows NT, Macintosh, VAX і в багатьох інших реалізаціях мови C++ для міні комп'ютерів дані типу int мають величину 32-біта (таку ж, як і дані типу long). При перенесенні програми C++ з одного середовища в інше можуть виникнути проблеми, пов'язані з різними величинами типів даних в різних реалізаціях. Але невеликі запобіжні засоби можуть звести цю проблему до мінімуму.

При оголошенні змінних імена даних цих типів використовуються точно так, як і імена даних типу int:

short score; //створює змінну типу short int

правильного значення після сьомої цифри "3". Змінна tub є правильною з точністю до сіми значущих цифр. (Ця система гарантує шість, правильних значущих цифр для даних типу float, але це у гіршому разі.) Проте значення змінної типа double, що відображається, містить 13 цифр "3". Отже воно є правильним з точністю щонайменше, до 13 значащих цифр. Це не повинне дивувати, оскільки даний тип даних гарантує 15 правильних значущих цифр. Окрім, того, слід звернути увагу на те, що якщо помножене на міліон значення змінної tub помножити ще на десять, то отриманий результат буде неточним; це знову вказує на обмежену точність даних типу float.

5.2. Константи з плаваючою крапкою

Коли використовувати в програмі константу з плаваючою крапкою, в якому форматі вона будет зберігатися в комп'ютері? За замовчуванням константи з плаваючою крапкою, наприклад 8.24 і 2.4E8, матимуть тип даних double. Якщо потрібна константа типу float, до неї додається суфікс f або F. Щоб тип константи був long double, додається суфікс l або L.

1.234f //константа типу float
2.45E20F // константа типу float
2.345324E28 // константа типу double
2.2L //константа типу long double

```
cout << 10 * million* tub << “\n”;
cout << “mint = “ « mint <<” and a million mints = ”;
cout << million * mint << “\n”;
return 0;
}
```

В результаті виконання програми, виходять наступний результат

tub =3.333333, a million tubs = 3333333.250000, and ten million tubs = =33333332.000000

mint =3.333333 and,a million mints = 3333333.333333

Примітки до програми

Об'єкт cout звичайно ігнорує праві нулі. Наприклад, він відобразить число 3333333.250000 як 3333333.25. Виклик методу coutisetf() відміняє цей режим, по крайній мірі в нових реалізаціях. Головне на що слідує звернути тут увагу, — це менша точність даних типу float в порівнянні з даними типу double. Змінні tub і mint ініціалізувалися одним і тим же значенням 10.0/3.0. Це дає в результаті число 3.33333333333333333... (і т.д.). Оскільки об'єкт cout виводить шість цифр праворуч від десяткової крапки, бачемо, що значення змінних tub і mint є точними. Проте, після того як вони множаються в програмі на мільйон, бачимо, що значення змінної tub відрізняється від


```
int temperature;        //створює змінну типу int
long position;          //створює змінну типу
long int
```

Насправді, short – це скорочення від short int, a long – скорочення long int, але навряд чи хто використовує розширені форми запису.

Дані цих трьох типів: int, short і long – є даними із знаком, а це означає, що діапазон їх можливих значень приблизно порівну між позитивними і негативними значеннями. Наприклад, 16-бітові дані типу int можуть лежати в діапазоні від -32768 до +32767.

Якщо вимагається взнати величини цілих чисел в конкретній системі, то це можна зробити за допомогою програмних інструментів, що є в мові С++. По-перше, операція **sizeof** повертає розмір даних будь-якого типа або розмір змінної (в байтах). Оператор (операція) – це елемент мови, за допомогою якої виконуються певні дії над одним або декількома елементами даних і в результаті виходить деяке значення. Наприклад, операція складання, представлена знаком +, викликає складання двох чисел. По-друге, заголовний файл **climits** (в більш ранніх реалізаціях – заголовний файл **limits.h)** містить інформацію про граничні значення цілочисельних даних різних типів. Зокрема, в

ньому визначені символічні імена для представлення різних граничних значень. В програмі limits.cpp з додатку 1 демонструється, як використовувати опис засобу. Там же ілюструється використовування ініціалізації, яка полягає в привласнення змінної деякого значення.

2.2. додаток 1. Програма limits.cpp

```
// limits.cpp – деякі граничні значення для цілих чисел
#include <iostream.h>
unsing namespace std;
#include <limits.h>
int main ()
{
        int  n_int = INT_MAX;      // ініціалізація змінної n_int максимальним значенням типу int
        short  n_ short = SHRT MAX;       // символи визначені в файлі limits.h
        long  n_long = LONG_MAX;
        // операція sizeof видає розмір даних будь-якого типу або змінної
        cout << “int is” << sizeof (int) << “bytes.\n”;
        cout << “ short is” << sizeof n_short << “bytes.\n”;
        cout << “ long is” << << sizeof n_long << “bytes.\n\n”;
```



```
#define DBL_MIN_10_EXP -307
#define FLT_MIN_10_EXP  -37
#define LDBL_MIN_10_EXP  -4931
```

В програмі з додатку 7 порівнюються типи даних **float** і **double** по критерію точності представлення чисел (мається на увазі число значущих цифр).

5.1. додаток 7. Програма floatnum.cpp

```
// Програма floatnum.cpp — типи даних з плаваючою точкою
#include <iostream.h>
using namespace std;
int main()
{
cout.setf  (ios_base::fixed,  ios_base::floatfiold);        // дані виводяться в
//режимі з фіксованою крапкою
float  tub  =  10.0  /.3.0;        // приблизно шість правильних, значущих цифр
double  mint  =  10.0/3.0;        //приблизно  15  правильних значущих цифр
const float million = 1.0e6;
cout << “tub = ” << tub;
cout  << “, a million tubs =” << million * tub;
cout << “, \nand ten million tubs = ”;
```

меншу, ніж дані типу double. Всі ці три типи даних можуть мати однакові розміри. Проте, як правило, дані типу float мають величину 32 біти, дані типу double — 64 біти і дані типу long double — 80, 96 або 128 бітів. Крім того, мінімальний діапазон значень показника для всіх трьох типів лежить в межах від -37 до + 37. Ви можете заглянути в заголовний файл **cfloat** або **float.h**, щоб взнати граничні значення для всіх типів даних з плаваючою крапкою в своїй системі. (Файл **cfloat** — це адаптована для мови C++ версія файлу **float.h** мови C.) Ось, наприклад, деякі елементи з файлу **float.h** для реалізації Borland C++ Builder з поясненнями:

```
// це мінімальне число значущих цифр
#define DBL_DIG 15         //double
#define FLT_DIG  6         // float
#define LDBL_DIG 18       // long double
//   це число бітів, що використовується для  представлення мантиси #define DBL_MANT_DIG      53
#define FLT_MANT_DIG      24
#define LDBL__MANT_DIG   64
// це максимальне і мінімальне  значення, показника
#define DBL_MAX_10_EXP   +308
#define FLT_MAX_10_EXP    +38
#define LDBL_MAX_10_EXP +4932
```



```
        cout << “Maximum values: \n”;
        cout << “int: “ << n_int << “\n”;
        cout << “short: “ << n_short << “\n”;
        cout << “long: “ << n_long << “\n”;

        cout << “Maximum int value = ”<< INT_MIN << “\n”;
        return 0;
}
```

В результаті виконання програми на платформі Microsoft Visual C++ 5.0 вийшли наступні результати:

```
int is 4 bytes.
short is 2 bytes.
long is 4 bytes.
Maximum values:
int: 2147483647
short: 32767
long: 2147483647
Minimum int value = -2147483648
```

Результати при виконанні програми на іншій платформі (Borland C++ 3.1 для DOS):

```
int is 2 bytes,
short is 2 bytes.
long is 4 bytes.
```

Maximum values:

int: 32767

short: 32767

long: 2147483647

Minimum int value = -32768

Примітки до програми

Оператор sizeof видає інформацію про те, що тип даних int має величину 4 байти в базовій системі, в якій використовується 8-бітовий байт. Оператор sizeof можна використовувати як з ім'ям типу даних, так і з ім'ям змінної. Коли оператор sizeof використовується з ім'ям типу даних, наприклад з int, ім'я береться в круглі дужки. Але, якщо оператор використовується з ім'ям змінної, наприклад з n_short, круглі дужки необов'язкові:

cout << "int is " << sizeof (int) << " **bytes . \n" ;**

cout << "short is " << sizeof n short << " **bytes.\n";**

В заголовному файлі limits.h визначені символічні константи, представляючі граничні значення для даних різних типів. Як вже згадувалося, константа INT_MAX представляє максимально можливе значення, яке може приймати тип даних int; в системі DOS це буде 32767. Виготівник поставляє з компілятором такий файл limits.h, в

5. Типи даних з плаваючою крапкою

В мові C++, як і в стандарті ANSI C, є три типи даних з плаваючою крапкою: float, double і long double. Дані цих типів характеризуються числом значущих цифр, які вони можуть мати, і мінімально допустимим діапазоном значень експоненти. *Значущі цифри* — це цифри, що визначають значення числа. Наприклад, коли висота гори Шаста (Shasta) в Каліфорнії записується у такому вигляді: 14162 фути, — то при цьому використовується п'ять значущих цифр, оскільки висота указується з точністю до фута. Але коли пишуть, що висота гори Шаста рівна приблизно 14000 футів, то при цьому використовуються дві значущі цифри, оскільки висота закруглена до тисяч футів. В цьому випадку три цифри, що залишилися, є просто заповнювачами розрядів числа. Кількість значущих цифр не залежить від положення десяткової крапки. Наприклад, можна написати, що висота рівна 14162 футів. Тут знову використовується п'ять значущих цифр, оскільки висота указується з точністю до 5-й цифри.

В результаті вимог, що пред'являються в мовах C і C++ до числа значущих цифр, дані типу float мають мінімальну величину 32 біти, дані типу double — мінімальну величину 48 бітів (але при цьому не меншу, ніж дані типу float), і дані типу long double мають величину, не

bool start = -100; //змінній start привласнюється значення true

bool stop = 0; // змінній stop привласнюється значення false

якому відображені значення, відповідні даному компілятору. Наприклад, для системи з 32-бітовими даними типу int у файлі limits.h визначена константа INT_MAX, рівна 2147483647.

Ініціалізація є об'єднанням привласнення значення з оголошенням. Наприклад, оператор

int n_int 5= INT_MAX;

оголошує змінну n_int і привласнює їй максимальне значення, можливе для типу даних int. Для ініціалізації значення змінних можна також використовувати стандартні константи. Ініціалізувати змінну можна, привласнюючи їй значення іншої змінної, при умові, що ця змінна була визначена раніше. Можна навіть ініціалізувати змінну значенням виразу за умови, що значення всіх компонентів виразу відомі під час компіляції;

int uncles = 5; //ініціалізує змінну uncles значенням 5

int aunts = uncles; //ініціалізує змінну aunts значенням 5

int chairs = aunts + uncles + 4 ; //ініціалізує змінну chairs значенням 14

Якщо оголошення змінної uncles перемістити в кінець цього списку операторів, то ініціалізації змінних aunts і chairs стануть недійсними, оскільки значення змінної

uncles не буде відоме в той час, коли компілятор намагатиметься проініціалізувати ці дві змінні.

2.3. Типи даних без знака

Для кожного з трьох тільки що розглянутих типів даних є його різновид: тип даних без знака. Дані цих типів не можуть приймати негативні значення. Перевагою тут є те, що збільшується максимально можливе значення даних. Наприклад, тип даних short представляє дані, що лежать в діапазоні від -32768 до +32767, тоді як різновид цього типу даних без знака може представляти дані, що лежать в діапазоні від 0 до 65535. Звичайно, типи даних без знака можна використовувати тільки для величин, які ніколи не бувають негативними і які торкаються, наприклад, населення, чисел в інвентаризаційній відомості і кількості опадів в місяць. Щоб створити різновид цілочисельної змінної без знака, просто модифікують оголошення за допомогою ключового слова unsigned:

```
unsigned short change;  //змінна типу unsigned short
unsigned int rovert;   //змінна типу unsigned int
unsigned quarterback;  //також змінна типа unsigned int
unsigned long gone;   //змінна типу unsigned long
```

Зверніть увагу, що ключове слово unsigned є скороченням від unsigned int.

4. Тип даних bool

Стандарт ANSI/ISO мови C++ добавив новий тип даних bool. Він названий в честь англійського математика Джорджа Буля (George Boole), який розробив математичне представлення законів логіки. В програмуванні змінна Boolean – це така змінна, яка може приймати два значення: true (истина) або false (ложь). В минулому в мові C++ , як і в мові C, дані типу Boolean були відсутні. Замість цього в мові C++ не нульові значення інтерпретувалися як значення true, а нульові значення – як значення false. Але тепер для представлення даних цих значень можна використовувати тип даних bool, також можна записувати оператори, подібні наступному:

```
bool isready = true;
```

Літерали true і false можуть бути перетворені в дані типу int шляхом підвищення типу, при цьому true перетвориться в 1, а false – в 0:

```
int ans = true;  //змінній ans привласнюється 1
int promise = false;  //змінній promise привласнюється 0
```

Крім того, будь-яке числове значення може бути перетворене в значення типу bool неявно. Будь-яке нульове значення перетвориться в значення true, а не нульове – в значення false:

даних типу **char** рівний 16 бітам або більший. По-друге, реалізація мови може допускати роботу як з основним, так і з розширеним набором символів. Звичайні 8-бітові дані типу **char** можуть використовуватися для представлення основного набору символів, а дані нового типу **wchar_t** — для представлення розширеного набору символів. Дані типу **wchar_t** є цілочисельними даними, що мають достатню величину для уявлення великого розширеного набору символів, використаного в даній системі. Дані цього типу мають таку ж величину і знак, що і дані одного з цілочисельних типів, який називається *базовим типом.* Вибір базового типу залежить від реалізації, і одній системі це може бути тип даних **unsignt short,** а в іншій — тип **short** або який-небудь інший.

В додатку 2 ілюструється використовування типів даних без знака.

2.4. додаток 2. Програма exceed.cpp

```
// exceed.cpp — робиться спроба вийти за межі діапазонів значень цілих чисел
«include <iostream>
using namespace std;
«define ZERO 0          // визначає, що символ ZERO має значення 0
«include <limits.h> // визначає константу INT_MAX як найбільше значення типу int
int main ()
{
        short sam = SHRT_MAX;        // ініціалізує змінну максимально
можливим значенням
        unsigned short sue = sam;    // все гаразд, якщо змінна sam вже визначена
        cout  << "Sam has " <<   sam << "dollars and Sue has” << sue;
        cout << " dollars deposited. \nAdd $1 to each account. \ nNow ";
        sam » sam + 1;
        sue = sue + 1;
```

```
    cout << "Sam has " << sam << " dollars and Sue has " <<
sue;
    cout << " dollars deposited .\nPoor Sam!\n";
    sam = ZERO;
    sue = ZERO;
    cout << "Sam has " <<   sam << " dollars and Sue has "
<<  sue;
    cout << " dollars deposited.\n";
    cout << "Take $1 from each account. \nNow ";
    sam = sam - 1;
    sue = sue - 1;
    cout << "Sam has " <<sam << " dollars and Sue has " <<
sue;
    cout << " dollars deposited. \nLucky Sue! \n" ;
    return 0;
}
```

В ньому також показано, що може відбутися, якщо в програмі буде зроблена спроба вийти за межі діапазону можливих значень для цілочисельних даних якого-небудь типу.

В результаті виконання програми виводяться наступні дані:

Sam has 32767 dollars and Sua has 32767 dollars deposited.

чином, то можна явно визначати їх як **signed char** або **unsigned char:**

```
char fodo;    //дані можуть бути як із знаком, так і без знака
unsigned char bar;  //це безумовно дані без знака
signed char snark;  //це безумовно дані із знаком
```

Ці відмінності особливо важливі, якщо у вас дані типу **char** є числовими даними. Тип даних **signed char** має діапазон значень від -128 до 127, а тип **unsigned char -** від 0 до 255. Припустимо, наприклад, що необхідно привласнити змінній типу **char** значення 200. В одних системах ця операція пройде успішно, а в інших ні. Проте для змінної типу **unsigned char** ця операція пройде успішно в будь-якій системі. З другого боку, якщо змінна типу char служить для зберігання стандартних символів коду ASCII, то неважливо, чи має тип даних **char** знак або не має; і в одному, і в іншому випадку можна використовувати змінну цього типу.

3.7. Тип даних wchar_t

Іноді програмам доводиться працювати з символьними даними, величини яких більше, ніж визначається 8-бітовим байтом; наприклад, японський набір символів (kanji). Справитися з такою ситуацією в мові C++ можна двома способами. По-перше, якщо великий набір символів є основним набором символів для даної реалізації, то постачальник компілятора може визначити, що розмір

```
    cout << "\aCode verified! Proceed with Flan Z3!\n";
    return 0;
}
```

Ця програма виводить на екран наступний текст:

Operation "HyperHype is now activated"! Enter your agent code:_____

Після друку символів підкреслення програма за допомогою символу повернення переміщає курсор назад на місце першого символу підкреслення. Тепер можна ввести секретний код і продовжити виконання програми. Ось остаточні результати:

Operation "HyperHype is now activated"!
Enter your agent code:42007007
You entered 42007007. . .
Code verified! Proceed with Plan Z3!

3.6. Типи даних signed char і unsigned char

На відміну від даних типу int, дані типу **char за** замовчуванням є ні даними із знаком, ні даними без знака. Право вибору залишається за розробниками реалізації мови C++ з тим, щоб цей тип даних краще відповідав конкретному апаратному забезпеченню. Якщо для вас життєво важливо, щоб дані типу char поводилися певним


```
Add $1 to each account.
Now Sam has -32768 dollars and Sue has  32768 dollars deposited.
Poor Sam!
Sam has 0 dollars and Sue has 0 dollars deposited.
Take $1 from each account.
Now Sam has -1 dollars and Sue has 65535 dollars deposited.
Lucky Sue!
```

В цій програмі змінній sam типа short і змінній sue типу unsigned short привласнюється найбільше значення для даних типу short, яке в нашій системі рівно 32767. Потім значення кожної змінної збільшується на одиницю. Для змінної sue це не викликає жодних проблем, оскільки нове значення набагато менше за максимально допустимий для цілого числа без знака. Але значення змінної sam змінюється з 32767 на -32768! Аналогічно цьому віднімання 1 з 0 не викликає жодних проблем відносно змінної sam, але значення беззнакової змінної sue змінюється з 0 на 65535. Як бачите, це багато в чому подібно тому, як змінюються значення спідометра автомобіля або лічильника відеомагнітофона. Якщо ви виходите за межу діапазону, то одне граничне значення замінюється іншим граничним значенням (з протилежного кінця діапазону). В мові C++ гарантується, що дані без знака поводяться саме таким

чином. Хоча в мові C++ не гарантується, що вихід за межі діапазону (переповнювання або втрата значущості) для цілочисельних даних із знаком відбуватиметься без видачі повідомлення про помилку, саме так найбільш часто працюють сучасні реалізації мови C++.

2.5. Цілочисельні константи

Цілочисельна константа— це константа, записана явно, наприклад: 212 або 1776. В мові C++, так само, як і в мові C, цілочисельні константи можуть записуватися в трьох системах числення: в десятковій системі числення (найпопулярніша система), вісімковій системі числення (фаворит системи UNIX старого випуску) і в шістнадцятковій системі числення (улюблена система хакерів, прагнучих отримати доступ до даних на апаратному рівні). Щоб визначити, до якої системи числення відноситься числова константа, в мові C++ використовується одна або дві перші цифри константи. Якщо перша цифра знаходиться в діапазоні 1-9, то число є десятковим; так, число 93 є десятковим. Якщо перша цифра рівна 0, а друга знаходиться в діапазоні від 1 до 7, то число є вісімковим; так, число 042 — це вісімкове число, еквівалентне десятковому числу 34, якщо першими двома символами є Ox або OX, то йдеться про шістнадцяткове число; так, число 0x42 — це шістнадцяткове число,

В програмі з додатку 6. демонструється декілька управляючих послідовностей. Символ попередження використовується, щоб привернути вашу увагу; символ нового рядка викликає просування курсора вперед; а символ повернення приводить до повернення курсора на одну позицію вліво. (Гудіні одного разу намалював картину з річкою Гудзон, використовуючи тільки управляючі послідовності; він, безумовно, був великим майстром в цій області.)

3.5. додаток 6. Програма bondini.cpp

```
// bondini.cpp — використовування управляючих послідовностей
#include <iostream>
using namespace std;
int main()
{
    cout << “\aOperation'\“HyperHype\” is now activated!\n”;
    cout << “Enter your agent code: _____\b\b\b\b\b\b\b\b”;
    long code;
    cin >> code;
    cout << “\aУои entered “ << code << ”...\n”;
```

Останній рядок дає такі вихідні дані:

Ben “Buggsie Hacker Was here”!

Зверніть увагу, що управляюча послідовність, наприклад \a, трактується як звичайний символ, наприклад, Q. Таким чином, для створення символьної константи управляюча послідовність береться в одинарні лапки, а усередині рядка одинарні лапки не потрібні;

В управляючих послідовностях можна використовувати вісімкові і шістнадцяткові коди символів. Наприклад, поєднанню клавіш Ctrl+Z відповідає код ASCII 26, який відповідає 032 у вісімковій системі і 0x1a — в шістнадцятковій системі числення. Цей символ може бути представлений будь-якою з наступних управляючих послідовностей: **\032** або **\0xla.** Взявши ці управляючі послідовності в одинарні лапки їх можна зробити символьними константами, наприклад, ‘\032’; можна також включати їх в рядок, наприклад, “hi\0x1a there”**.**

Якщо ви стоїте перед вибором — використовувати числову управляючу послідовність або символічну, наприклад \0x8 або \b, — краще використовуйте символічний запис. Числовий запис прив'язаний до конкретного коду, наприклад, до коду ASCII, а символічний запис справедливий при будь-якому коді і більш зрозумілий.

еквівалентної десятковому числу 66. В шістнадцяткових числах символи a-f і A-F позначають відповідно шістнадцяткові цифри 10—15. Шістнадцяткове число OxF еквівалентно десятковому числу 15, а 0xA5 — десятковому числу165 (10 помножити на 16 і додати 5, помножене на 1). Програма з додатку 3 була написана спеціально, щоб продемонструвати використовування цих систем числення.

2.6. додаток 3. Програма hexoct.cpp

```
// hexoct.cpp — демонструє використовування шістнадцяткових і вісімкових констант
#include <iostream>
using namespace std;
int main ()
{
        int chest = 42;       //десяткова цілочисельна константа
        int waist =0x42;      //шістнадцяткова цілочисельна константа
        int inseam =042; //вісімкова  цілочисельна константа
        cout << “Monsieur cuts a striking figure!\n”;
        cout << “chest = “” << chest  <<“ \n”;
        cout << “waist = ” <<waist <<“\n”;
        cout << “inseam = ” << inseam << “\n”;
```

```
    return 0;
}
```

За замовчуванням об'єкт cout відображає на екрані цілі числа в десятковому вигляді, незалежно від того, як вони записані в програмі; це демонструють наступний результат виконання програми:

```
Monsieur cuts a striking figure!
 chest =42
 waist =66
inseam = 34
```

Всі ці системи числення використовуються тільки для зручності. Наприклад, якщо ви прочитали, що сегмент відеопам'яті CGA еквівалентний шістнадцятковому числу B000, то вам не вимагається перетворювати його в десяткове число 45056 перед використовуванням в своїй програмі. Ви можете просто записати OxBOOO. Але незалежно від того, в якому вигляді ви записуєте число 10: як 10, 012 або 0xA — воно зберігається в комп'ютері одним і тим же способом: як двійкове число.

‘a’ відповідає 97, коду ASCII для букви “a”

‘5’ відповідає 53, коду ASCII для цифри “5”

‘ ’ відповідає 32, коду ASCII для символу пропуск

‘!’ відповідає 33,коду ASCII для знаку оклика

Краще використовувати такий запис, ніж явно записувати числові коди. Він зрозуміліший і не прив'язаний до конкретного коду. Якщо в системі використовується код EBCIDIC, то 65 не буде кодом для символу A, але запис ‘A’ завжди представлятиме даний символ.

Деякі символи не можна вводити в програму з клавіатури безпосередньо. Наприклад, символ нового рядка не можна ввести, просто натискуючи клавішу Enter; програма текстового редактора інтерпретує натиснення цієї клавіші як вимога почати новий рядок у вашому файлі початкового коду. Труднощі із записом інших символів відбуваються тому що в мові C++ їм надається особливе значення. Наприклад, символ подвійних лапок обмежує рядки, тому його не можна узяти і просто вставити в середину рядка.

Таку форму запису можна використовувати в рядках або символьних константах:

```
char  alarm =  '\a';
cout  <<  alarm <<  “Don't  do  that  again!\a\n”;
cout <<  “Ben  \“Buggsie\”  Hacker was  here!\n”;
```

cout << '$';

виводив не символ $, а його код ASCII. Але оператор

cout.put ('$');

виводив символ так, як було потрібно. В подальших версіях мови C++ (що з'явилися після версії 2.0) символьні константи зберігаються як дані типу char, а не int. Тому в них об'єкт cout обробляє символьні константи правильно. Як символ нового рядка в мові C++ завжди можна було використовувати рядок "\n". Тепер для цієї мети можна також використовувати символьну константу '\n':

cout << "\n"; //використовується рядок

cout << '\n'; //використовується символьна константа

Рядок укладений не в одинарні, а в подвійні лапки і може містити більше одного символу. Рядки, навіть що складаються з одного символу, не є даними типу char.

3.4. Константи типу char

В мові C++ символьні константи можна записувати декількома різними способами. Найпростіший спосіб запису звичайних символів, наприклад букв, цифр, розділових знаків, – узяти символ в одинарні лапки! Цей запис замінює числовий код символу. Наприклад, в системі, що включає код ASCII, є наступні відповідності:

'A' відповідає 65, коду ASCII для букви "A"

3. Тип даних char: символи і малі цілі числа

Тип даних char призначений для представлення символів, наприклад, букв і цифр. Але якщо зберігання чисел не є для комп'ютерів великою проблемою, то зберігання символів — це зовсім інша справа. В мовах програмування був знайдений простий вихід: для представлення символів використовується числовий код. Таким чином, тип даних char — це ще один тип цілочисельних даних. Гарантується, що дані цього типу мають достатні розміри для представлення всього діапазону основних символів — всіх букв, цифр, знаків і т.п. — обчислювальної системи, для якої призначена реалізації мови. На практику в більшості систем основний набір символів містить менше 256 символів, тому для його представлення достатньо одного байта. Тому, хоча тип даних char найчастіше всього використовується для представлення символів, його також можна використовувати для представлення цілих чисел, менших, ніж числа типу short.

Найпоширенішим набором символів США є набір, символів ASCII, описаний в додатку С. Символи цього набору представлені числовим кодом (кодом ASCII). Наприклад, 65 — це код символу A. В реалізації C++ використовується будь-який код, який є природнім для

даної обчислювальної системи, наприклад, код EBCDIC на великих ЕОМ фірми IBM. Ні код ASCII, ні код EBCDIC не можуть обслуговувати потреби програмістів у світовому масштабі, і в мові C++ є розширений символьний тип даних, який може представляти більший діапазон, значень.

Робота з даними типу char демонструється в додатку 4.

3.1. додаток 4. Програма chartype.cpp

```
chartype.cpp — застосування даних типу char
#lnclude <iostream>
using namespace std;
int main( )
{
        char ch;    //оголошення змінної типу char
        cout << "Enter a character: \n";
        cin >> ch;
        cout << "Holla";
        cout << "Thank you for the" << ch << "character.
\n";
        return 0;
}
```


тільки окремими об'єктами свого класу; в даному випадку це об'єкт cout. Щоб використовувати функцію-елемент класу при роботі з таким об'єктом, як cout, ім'я об'єкту (cout) і ім'я функції (put()) об'єднуються з допомогою, крапки. Ця крапка називається оператором *приналежності, Запис* cout.put() означає, що функція-елемент put() деякого класу використовуватиметься об'єктом cout() цього ж класу.

Функція-елемент cout.put() є альтернативою операції <<, що використовується для відображення символу. В цьому місці можна здивуватися: навіщо взагалі потрібна функція cout.put()? Відповідь здебільшого лежить в історичній площині. До появи версії 2.0 мови C++ об'єкт cout відображав символьні змінні як символи, а символьні константи, наприклад 'M' і '\n' — як числа. Проблема полягала в тому, що в перших версіях мови C++, як і в мові C, символьні константи зберігалися як дані
типу int. Таким чином, код 77 для символу 'M' зберігався в 16- або 32-бітовому елементу пам'яті. А змінні типу char займали, як правило, 8 бітів. Такий оператор, як
char c = 'M';
копіював 8 бітів (8 значущих бітів) константи 'M' в змінну c. Нажаль це означало, що константа 'M' і змінна c були для об'єкту cout абсолютно різними величинами, хоча і мали одне і те ж значення. Тому такий оператор як

Оскільки змінна c насправді містить ціле число, по відношенню до неї можна виконувати операції для цілих чисел, наприклад, надбавка одиниці. В результаті змінна c матиме значення 78. Потім це нове значення привласнюється змінній i. (Можна було просто додати 1 до змінної i.) Об'єкт cout знову відображає це значення типу char як символ і значення типу int як число.

В мові C++ символи представлені у вигляді цілих чисел, що дуже зручно, оскільки спрощується виконання дій із значеннями символів. Не потрібно користуватися незручними функціями перетворення символів в код ASCII і назад.

Нарешті, в програмі використовується функція cout.put() для відображення як змінної c, так і символьної константи.

3.3. Функція-елемент: cout.put()

Але що є функція cout.put() і чому в її назві присутня крапка? Функція cout.put() знайомить з таким важливим поняттям об'єктно-орієнтованого програмування, як *функція-елемент.* Клас, визначає спосіб представлення даних і операції, які можна виконувати над цими даними. Функція-елемент належить класу і описує метод виконання дій з даними цього класу. В класі ostream, наприклад, присутня функція-елемент put(), призначена для виводу символів. Функції-елементи можуть використовуватися

Як завжди, запис \n в мові C++ означає символ нового рядка. В результаті виконання програми виходить наступний результат:

Enter a character:

M

Holla! Thank you, for the M character.

Цікаво тут те, що ви вводите символ M, а не відповідний код символу 77. І програма також друкує символ M, а не код 77. Проте, якщо ви "заглянете" в пам'ять, то знайдете, що в змінній ch зберігається значення 77. Це чудове перетворення пов'язано не з даними типу char, а з об'єктами cin і cout. Ці цікаві конструкції виконують, перетворення від вашого імені. При введенні даних об'єкт, cin перетворить вхідні дані, що є результатом натискання клавіші M, в значення 77. При виведені даних об'єкт cout перетворить значення 77 в символ M; функціонування об’єктів cin і cout визначає тип змінної. Якщо те ж саме значення 77 привласнити змінній типу int, то об'єкт cout, відобразить його як 77 (тобто об'єкт cout відобразить два символи 7). Цей момент, ілюструється в програмі з додатку 5. В ній також показано, як в мові C++ визначаються символьні константи: символ полягає в одинарні лапки, наприклад, 'M'; (Зверніть увагу, що в даному прикладі програми не використовуються подвійні лапки). В мові C++

одинарні лапки застосовуються, для завдання символів, а подвійні лапки — для завдання рядків. Об'єкт cout може обробляти і символи, і рядки, але це дві абсолютно різні речі.) І нарешті, в програмі застосовується, нова властивість об'єкту cout — функція cout.put(), яка відображає одинарний символ.

3.2. додаток 5. Програма morechar.cpp

```
//morechar.cpp — зіставлення даних типу char і типу int
#include <iostream>
using namespace std;
int main()
{
        char c = 'M';      //змінній привласнюється код ASCII для символа M
        int i = c;    //той же код привласнюєтьсязмінній типу int
        cout << “ASCII code for ” << c << “ is”  << A <<  “\n”;
        cout << “Add one to character code:\n”;
        c = c + 1;
        i = c;
        cout << “ASCII code for ” << c << “ is ” << i ‘\n’;
// використовування функції cout.putO для  відображення константи типу char
```



```
        cout << “Displaying char c using cout.put(c): ”;
        cout.put (c) ;
//використовування функції cout.put() для  відображення константи типу char cout.put(‘!’);
        cout << “\nDone\n”;
        return   0;
}
```

В ході виконання програми виходить наступний результат:

```
The ASCII code for M is 77
Add one to the character code:
The ASCII code for N is 78
Displaying char c using cout.put(c): N!
Done
```

Примітки до програми

Запис ‘M’ представляє числовий код для символу M, тому ініціалізація змінної з типу char значенням ‘M’ означає занесення в неї числа 77. Потім ідентичне значення привласнюється змінній і типу int. Тепер обидві змінні, c і *i*, містять число 77, Потім об'єкт cout відображає змінну c як символ M і змінну і як число 77. Як вже мовилося раніше, тип значення визначає функціонування об'єкту cout, коли він вирішує, яким чином відображати дане значення. Це ще один приклад того, що cout — “розумний об'єкт”.