**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: Типы данных и их внутреннее представление в памяти.

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. |  | Адигюзалова А.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Изучение типов данных и их представления в памяти; получение практических навыков работы с различными типами данных и управляющими структурами и инструкциями.

**Основные теоретические положения.**

Тип данных— это характеристика набора данных, которая определяет диапазон его возможных значений, список допустимых операций, а также способ хранения набора данных. Любые данные хранятся в памяти компьютера в виде двоичных кодов.

Типы данных можно разделить на две группы: простые и структурированные типы. Простые типы данных представляют неразделимые данные, не имеющие внутренней структуры (это, например, числа, символы и т.д.). Структурированные типы данных имеют внутреннюю структуру. Структурированные типы строятся на основе простых типов данных.

Другой уровень классификации разделяет все типы данных на предопределенные (изначально встроенные в язык программирования) и пользовательские (типы данных, определяемые программистом) типы данных.

Основные (предопределенные) типы данных часто называют арифметическими, так как их можно использовать в арифметических операциях. Для их описания определены следующие ключевые слова:

* int – целый тип данных;
* float – вещественный тип данных;
* double – вещественный тип данных с двойной точностью;
* bool – логические тип данных;
* char – символьный тип данных.

Типы int, bool и char относят к группе целочисленных типов, а float и

double – к группе вещественных типов.

Существует четыре спецификатора типа, которые уточняют их внутреннее представление и диапазон значений стандартных типов:

* short – короткий;
* long – длинный;
* signed – знаковый;
* unsigned – без знаковый;

Допустимы не все сочетания спецификаторов и типов данных (так, unsigned double является недопустимым сочетанием, short int - допустимым).

Переменная — это именованная область памяти, в которой хранятся данные определенного типа. Каждая переменная имеет имя и значение. Общее правило определения переменной можно сформулировать так:

**[класс памяти] <тип данных> <идентификатор - имя> [инициализатор];**

Константа – это величина, значение которой в процессе работы программы не изменяется. Константы бывают двух видов: константы – литералы и именованные константы. Константы – литералы представляют собой сами значения; именованные константы задаются с помощью ключевого слова const.

Размер типа int не определяется стандартом, а зависит от компьютера и компилятора. Для 16-разрядного процессора под величины этого типа отводится 2 байта – в этом случае диапазон возможных значений составляет -32 768 ... 32 767 (2 в степени 16 различных значений). Для 32-разрядного - 4 байта – диапазон значений -2 147 483 648 ... 2 147 483 647 (2 в степени 32 различных значений).

Есть три типа данных для хранения вещественных значений: float, double и long double. Все эти типы предназначены для представления отрицательных и положительных значений (спецификатор **unsigned**к ним не применим) в разных диапазонах:

* тип **float**занимает в памяти 4 байта с диапазоном абсолютных значений от 3.4е-38 до 3.4е+38;
* тип **double** занимает в памяти 8 байт с диапазоном абсолютных значений от 1.7е-308 до 1.7е+308;
* тип **long double**занимает в памяти 10 байт с диапазоном абсолютных значений от 3.4e-4932 до 3.4e+4932.

Вещественные типы данных (типы данных с плавающей точкой) хранятся в памяти компьютера иначе, чем целочисленные. Внутреннее представление вещественного числа состоит из двух частей — мантиссы и порядка. Тип **float** занимает 4 байта, из которых один двоичный разряд отводится под знак мантиссы, 8 разрядов под порядок и 23 под мантиссу. Для величин типа **double**, занимающих 8 байт, под порядок и мантиссу отводится 11 и 52 разряда соответственно. Длина мантиссы определяет точность числа, а длина порядка — его диапазон.

Величины логического типа могут принимать только значения **true** и **false**, являющиеся зарезервированными словами. Внутренняя форма представления значения **false** - О (нуль). Любое другое значение интерпретируется как **true**. При преобразовании к целому типу **true** имеет значение **1**(единица). В памяти переменные этого типа занимают 1 байт.

Для обозначения символьного типа данных используется ключевое слово **char**.

Под величину символьного типа отводится количество байт, достаточное для размещения любого символа из набора символов для данного компьютера, что и обусловило название типа. Как правило, это 1 байт. Тип **char**, как и другие целые типы, может быть со знаком или без знака. В величинах со знаком можно хранить значения в диапазоне от **-128** до **127**. По умолчанию тип **char** является знаковым, то есть спецификатор **signed** использовать не обязательно. При использовании спецификатора **unsigned** значения могут находиться в пределах от **0** до **255**. Этого достаточно для хранения любого символа из 256-символьного набора ASCII. Величины типа **char** могут применяться и для хранения целых чисел, не выходящих за границы указанных диапазонов. Спецификаторы **short** и **long** к этому типу данных не применяются.

Обработка данных выполняется с помощью операций. **Операция** – это действие, осуществляемое над операндами.

В зависимости от количества операндов операции бывают унарными (с одним операндом), бинарными (с двумя операндами) и одна тернарная (с тремя операндами).

Управляющие структуры используются для управления ходом выполнения программы. Существует три категории управляющих инструкций:

* Инструкции выбора. К ним относят условную инструкцию **if** и инструкцию множественного выбора **switch**;
* Итерационные инструкции. К ним относят цикл с предусловием **while**, цикл с постусловием **do while** и итерационный цикл **for**;
* Инструкции перехода. К ним относят прекращение выполнения циклических инструкций и инструкции switch **break**, переход к следующей итерации цикла **continue**, прекращение выполнения функции **return**, переход по метке **goto**.

Условная конструкция if-else направляет ход программы по одному из возможных путей в зависимости от условия. Она проверяет истинность условия, и если оно истинно, выполняет блок инструкций. В простейшем виде конструкция if имеет следующую сокращенную форму:

if (условие)

{

    инструкции;

}

Цикл for имеет следующее формальное определение:

for (инициализатор; условие; итерация)

{

    // тело цикла

}

Инструкция множественного выбора **switch** служитдля ветвления программы во многих направлениях:

switch (<Выражение>)

{

case <Константа 1>:

<Последовательность инструкций 1>

break;

case < Константа 2>:

< Последовательность инструкций 2>

break;

……

case < Константа N>:

< Последовательность инструкций N>

break;

default:

< Последовательность инструкций>

}

**Массив** представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов. Все массивы можно разделить на две группы: одномерные и многомерные. Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных. Объявление в программах одномерных массивов выполняется в соответствии со следующим правилом:

**<Базовый тип элементов> <Идентификатор массива>[<Количество элементов>]**

Так, например:

int ArrInt [10], A1 [20];

Обращение к определенному элементу массива осуществляется с помощью указания значения индекса этого элемента:

A1 [4] = -1200;

cout << A1 [4];

При обращении к конкретному элементу массива этот элемент можно рассматривать как обычную переменную, тип которой соответствует базовому типу элементов массива, и осуществлять со значением этого элемента любые операции, которые характерны для базового типа.

При объявлении массива его можно инициализировать определенными значениями:

short S [5] = {1, 4, 9, 16, 25};

Или:

short S [] = {1, 4, 9, 16, 25};

**Постановка задачи.**

Необходимо разработать алгоритм и написать программу, которая позволяет:

1. Вывести, сколько памяти (в байтах) на компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.
2. Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.
3. Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.
4. Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

**Выполнение работы.**

Код программы представлен в приложении А.

Описание кода и использованных алгоритмов:

При запуске программы перед пользователем появляется окно, в котором на экран выводится

1. Количество байт памяти, которое отводится под различные типы данных.
2. Интерактивное поле, где пользователь может задать число типа integer, двоичный код которого он хочет увидеть. После ввода пользователем данных, на экран выводится двоичное представление введенного числа в памяти компьютера.
3. Интерактивное поле, где пользователь может задать число типа float, двоичный код которого он хочет увидеть. После ввода пользователем данных, на экран выводится двоичное представление введенного числа в памяти компьютера.
4. Интерактивное поле, где пользователь может задать число типа double, двоичный код которого он хочет увидеть. После ввода пользователем данных, на экран выводится двоичное представление введенного числа в памяти компьютера.

Работа алгоритма и вывод на экран:

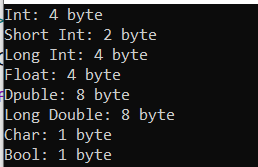


Рис. 1. Вывод на экран количества байт, отведенных под различные типы данных



Рис.2 Поле для ввода числа типа integer.



Рис. 3. Вывод на экран введенного ранее числа типа integer в двоичном виде (в данном случае было введено число 127)



Рис. 4. Поле для ввода числа типа float.



Рис. 5. Вывод на экран введенного ранее числа float в двоичном виде (в данном случае было введено число 3.14)



Рис. 6. Поле для ввода числа типа double.



Рис. 7. Вывод на экран введенного ранее числа double в двоичном виде (в данном случае было введено число 3.14)

Тестовые данные:

|  |  |  |
| --- | --- | --- |
| Int | Float | Double |
| 127, 23, -127 | 3.14, 2.89, -9.86 | 7.895, 3.14, -4.456 |

**Выводы.**

В ходе практической работы были изучены различные типы данных, а также понимание того, как выглядит их представление в памяти компьютера. Были получены практические навыки работы с типами данных и управляющими структурами и инструкциями языка C++. Была разработана программа, которая позволила наглядно увидеть представление таких типов данных как int, float и double в памяти компьютера.

Приложение А

рабочий код

#include <iostream>

using namespace std;

int main()

{

cout << "Int: " << sizeof(int) << " byte" << endl;

cout << "Short Int: " << sizeof(short int) << " byte" << endl;

cout << "Long Int: " << sizeof(long int) << " byte" << endl;

cout << "Float: " << sizeof(float) << " byte" << endl;

cout << "Dpuble: " << sizeof(double) << " byte" << endl;

cout << "Long Double: " << sizeof(long double) << " byte" << endl;

cout << "Char: " << sizeof(char) << " byte" << endl;

cout << "Bool: " << sizeof(bool) << " byte" << endl;

int intNumber;

cout << "Enter any integer number: ";

cin >> intNumber;

unsigned int intMask = 1 << (sizeof(int) \* 8 - 1);

for (int i = 1; i <= sizeof(int) \* 8; i++)

{

putchar(intNumber & intMask ? '1' : '0');

intMask >>= 1;

if (i % 8 == 0 || i == 1) {

cout << " ";

}

}

cout << endl;

cout << "Enter any float number: ";

unsigned int floatMask = 1 << (sizeof(int) \* 8 - 1);

union {

float floatNumber;

int placeHolder;

};

cin >> floatNumber;

for (int j = 1; j <= sizeof(float) \* 8; j++)

{

putchar(placeHolder & floatMask ? '1' : '0');

floatMask >>= 1;

switch (j) {

case 1:

cout << " ";

break;

case 9:

cout << " ";

break;

}

}

cout << endl;

cout << "Enter any fractional number: ";

unsigned int doubleMask = 1 << (sizeof(int) \* 8 - 1);

union {

double doubleNumber;

int arr[2];

};

cin >> doubleNumber;

for (int z = 1; z >= 0 ; --z)

{

for (int k = 1; k <= sizeof(int) \* 8; k++) {

putchar(arr[z] & doubleMask ? '1' : '0');

doubleMask >>= 1;

if ( z == 1 & (k == 1 || k == 12)) {

cout << " ";

}

}

doubleMask = 1 << (sizeof(int) \* 8 - 1);

}

}