ФГБОУ ВО “Чувашский государственный университет им.

И.Н. Ульянова”

Кафедра вычислительной техники

Лабораторная работа №1

**Типы данных языка С++.**

Вариант 9.

Выполнил: студент гр. Ивт-41-22

Иванов В.С.

Проверил: кандидат технических наук

Обломов И.А

Чебоксары 2023

Цель работы: изучить основные типы данных языка С++, научиться работать над основными типами данных.

Ход работы.

Основные (стандартные) типы часто называют арифметическими, поскольку их можно использовать в арифметических операциях. Для описания основных типов используют следующие ключевые слова: int (целый), char (символьный), wchar\_t (расширенный символьный), bool (логический), float (вещественный), double (вещественный с двойной точностью). Первые четыре типа называют целочисленными, т.к. они представляются в машине с помощью целых чисел. Два последних типа называют типами с плавающей точкой. Коды, формируемые компилятором для обработки целочисленных и вещественных данных, будут различными.

Существует четыре спецификатора типа, уточняющих внутреннее представление и диапазон значений стандартных типов: short (короткий), long (длинный), signed (знаковый), unsigned (беззнаковый).

***Целый тип (int).*** Размер типа int стандартом языка не регламентирован, он зависит от реализации компилятора и разрядности процессора.

Спецификатор short перед именем типа указывает компилятору, что под число требуется отвести 2 байта, независимо от разрядности процессора. Спецификатор long требует 4 байта.

Использование спецификатора signed (по умолчанию) предполагает, что старший разряд числа интерпретируется как знаковый (0- для положительных чисел, 1- для отрицательных). Спецификатор unsigned позволяет представлять только положительный числа. В зависимости от спецификаторов перед типом int диапазон представления различен.

***Символьный тип (char).*** Как правило под данные типа char выделяется 1 байт. Этого достаточно для хранения до 256 различных символов кода ASCII. Тип char, как и другие целые типы может быть со знаком или без знака.

Величины типа char могут использоваться для хранения целых чисел в указанных диапазонах.

***Расширенный символьный тип (wchar\_t).*** Тип wchar\_t предназначен для хранения символов, для которых 1 байта не достаточно, например, для символов в кодировке Unicode. Размер этого типа соответствует типу short ( 2 байта).

***Логический тип (bool).*** Множество значений этого типа – это true (1) и false (0). Любое другое значение интерпретируется как true. При преобразовании к целому типу true имеет значение 1, а false – 0.Для величин этого типа компилятор выделяет 1 байт.

***Типы с плавающей точкой (float, double, long double).*** Типы данных с плавающей точкой хранятся в памяти компьютера иначе, чем целочисленные. Под тип float обычно выделяется 4 байта, один разряд из которых отводится под знак мантиссы, 8 разрядов под порядок и 23 – под мантиссу. Мантисса – это число, большее 1 и меньшее 2.

Под величины типа double выделяется 8 байт. Спецификатор long перед double гарантирует выделение 10 байт.

Константы с плавающей точкой имеют по умолчанию тип double.

***Тип void.*** Кроме перечисленных стандартных типов, к основным типам относят тип void, множество значений которого пусто. Он используется для определения функций, не возвращающих никакого результата, а также для представления пустого списка параметров. Кроме того этот тип используется как базовый тип для указателей.

***Составные типы данных.***

Как уже было отмечено, к составным типам относят массивы, перечисления, структуры, указатели и ссылки, объединения и классы.

***Переименование типов(typedef).*** В некоторых случаях типу можно дать новое имя. Это позволяет сделать текст программы более ясным. Общий формат преобразования можно представить следующим образом:

typedef тип новое\_имя [ размерность ];

В данном контексте квадратные скобки и содержимое между ними следует рассматривать как необязательный элемент конструкции.

***Перечисления(enum).*** В некоторых случаях необходимо иметь конечное множество именованных констант, имеющих различные значения. Для этого удобно воспользоваться перечислимым типом данных. Формат перечислимого типа следующий:

enum [имя\_типа] {список\_констант};

Необязательное имя типа требуется в случае, если требуется определить переменные этого типа.

Структуры (struct). В отличие от других типов данных, например массивов, представляющих величины одного типа, структуры могут содержать в себе элементы различных типов. В языке С++ структура является видом класса и обладает всеми его свойствами.

Имя структуры (необязательное, при этом такая структура называется анонимной) представляет новый тип, определенный пользователем. Этот тип в дальнейшем может быть использован наравне со стандартными. При этом компилятор не дает никаких привилегий стандартным типам. Если имя структуры отсутствует, должен быть указан список описателей переменных, указателей или массивов.

Элементы структуры, именуемые полями, могут имеет любой тип, кроме типа определяемой структуры, но могут быть указателями на этот тип.

***«Операции» над типами данных.***

Слово операции в названии заключено в скобки, поскольку операции определены над данными определенных типов, а не над самими типами. Язык С++, являясь объектно-ориентированным, не предполагает, что стандартный тип или тип объявленный пользователем в свою очередь является объектом (переменной) какого-нибудь типа.

Однако, для работы с типами С++ предоставляет следующие возможности:

- определить размер типа в байтах оператором sizeof, в качестве аргумента может быть имя типа, имя переменной или более сложное выражение;

- задать альтернативное имя (синоним) существующему типу оператором typedef;

- средства преобразования одних типов в другие;

- механизм определения типа объекта в реальном времени (RTTI);

***Оператор sizeof.***

Оператор **sizeof**- это унарный оператор, возвращающий длину в байтах переменной или типа, помещенных в скобки.

В качестве аргумента операции может выступать любой тип, как стандартный, так и определенный пользователем, а также объект или переменная любого типа.

Этот оператор может использоваться при выполнении программы на машинах с процессорами, имеющими различную разрядность, например, 16 и 32. Это позволяет создавать машинно-независимые программы.

С помощью оператора sizeof можно определить размер поля структуры или класса.

***Преобразование типов.*** В языке С++ возможно несколько способов преобразования данных одного типа в другой, если это допустимо. Один из способов состоит в указании необходимого типа, заключенного в скобки перед данным другого типа, в двух формах: тип(выражение) или тип(выражение).

***Механизм определения типа объекта в реальном времени (RTTI).*** Этот механизм позволяет определить, на какой тип в текущий момент времени ссылается указатель. Для доступа к RTTI в стандарт языка введена операция typeid и класс type\_info. Класс содержится в заголовочном файле <typeinfo>.

Основная операция typeid, которая применима к основным и производным типам. Метод name возвращает указатель на строку, представляющую имя типа, а метод before выполняет побуквенное сравнение двух имен типов.

**Выполнение работы**.

Задание:

Открыть пустой проект в среде Visual Studio, набрать программу, содержащую объявление переменных основных типов, а также структуру, состоящую из 3-4 полей.

1. Определить число байт, необходимых для хранения основных типов для данной реализации компилятора и разрядности процессора. По возможности выполнить программу на машинах с разной разрядностью и оценить полученные результаты.
2. Осуществить преобразование отдельных типов с целью улучшения читаемости программы, а также сокращения длинных имен типов.
3. Объявить переменные перечислимого типа без инициализации констант и с их инициализацией. Выполнить допустимые для них операции. Определить число байт, требуемое для хранения таких переменных.
4. Объявить структуру. Определить число байт, требуемое для хранения всех полей структуры. Оценить полученные результаты и сопоставить с результатами по пункту 1.Определить число байт, необходимых для хранения отдельных полей.
5. Пользуясь преобразованием типов, осуществить преобразования объектов к другим типам. Оценить возможность или невозможность преобразований.
6. Пользуясь механизмом определения типа в реальном времени, определить тип фактических объектов и выражений.

По результату выполнения задач был написан код:

#include <iostream>

using namespace std;

int integer = 2;

char Char = 'C';

wchar\_t Wchar = L'A';

bool Bool = true;

float pi = 3.14;

double pi1 = 3.141592;

struct Structure {

int StructureNumber;

char StructureFirstChar;

int StructureElement;

}Structure;

int main() {

setlocale(0,"");

cout << "1 Пункт:" << endl;

cout << "число байт, необходимых для хранения целого типа данных (int) : " << sizeof(integer) << endl;

cout << "число байт, необходимых для хранения символьного типа данных (char) : " << sizeof(Char) << endl;

cout << "число байт, необходимых для хранения расширенного символьного типа данных (wchar\_t) : " << sizeof(Wchar) << endl;

cout << "число байт, необходимых для хранения логического типа данных типа данных (bool) : " << sizeof(Bool) << endl;

cout << "число байт, необходимых для хранения целого типа данных (float) : " << sizeof(pi) << endl;

cout << "число байт, необходимых для хранения целого типа данных (double) : " << sizeof(pi1) << endl;

cout << "число байт, необходимых для структуры с тремя полями (struct) : " << sizeof(Structure) << endl;

cout << "2 Пункт:" << endl;

typedef unsigned long ULONG;

ULONG A = 3896893782;

cout << "3 Пункт:" << endl;

enum Human1;

enum Human{Australian, Russian, French};

Human human = Russian;

if (human == Russian) {

cout << "Human is Russian" << endl;

}

cout << "Число байт, требуемое для хранения: " << sizeof(Human) << endl;

cout << "4 Пункт:" << endl;

cout << "Число байт, требуемое для хранения первого поля:" << sizeof(Structure.StructureNumber) << ", второго поля:" << sizeof(Structure.StructureFirstChar) << ", третьего поля:" << sizeof(Structure.StructureElement) << endl;

cout << "5 Пункт:" << endl;

pi1 = (int)pi1;

cout << pi1 << endl;

pi = (int)pi;

cout << pi << endl;

integer = (float)integer;

cout << integer << endl;

cout << "6 Пункт:" << endl;

cout << "Имя типа - " << typeid(integer).name() << endl;//int

cout << "Имя типа - " << typeid(Char).name() << endl;//char

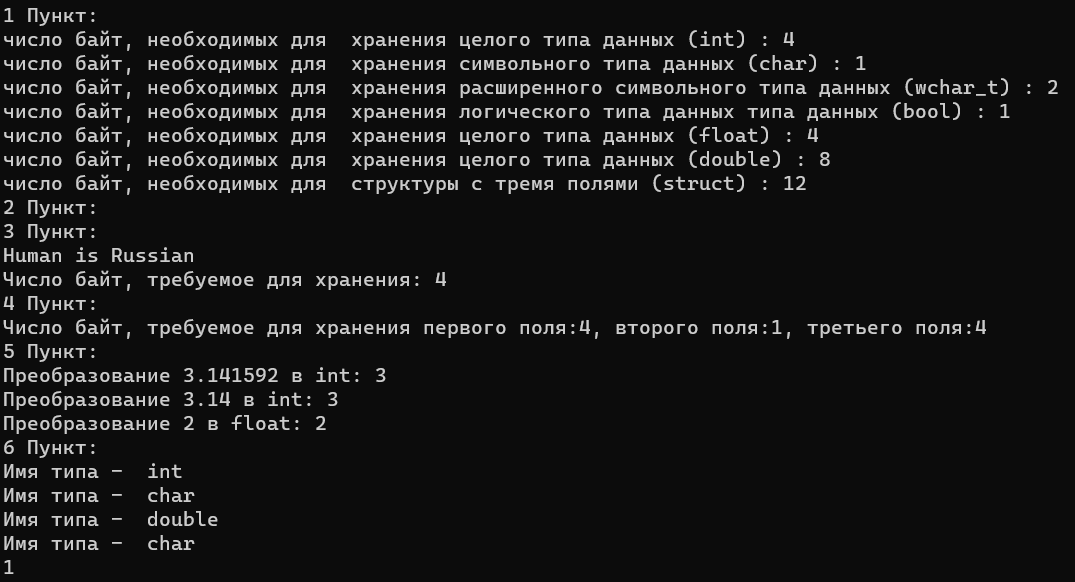
cout << "Имя типа - " << typeid(3.14+3.141592).name() << endl;//double

cout << "Имя типа - " << typeid('C').name() << endl;//char

cout << typeid(integer).before(typeid(pi)) << endl;// true

}

Результат работы программы:



Вывод: изучил основные типы данных языка С++, научился работать над основными типами данных