ФГБОУ ВО “Чувашский государственный университет им. И. Н. Ульянова”
Факультет: ИВТ
Кафедра: Вычислительной техники
Предмет: Объектно-ориентированное программирование

Лабораторная работа №1

Типы данных языка C++

Выполнил: студент группы ИВТ-41-20
Галкин Дмитрий
Проверил: Обломов И.А.

Чебоксары 2021 г.

Индивидуальное задание

1. Число байт, необходимое для хранения основных типов данных:
   - int – 4 байта
   - char – 1 байт
   - bool – 1 байт
   - float – 4 байта
   - double – 8 байт
   - wchar_t – 4 байта

2. Преобразование типов осуществляется с помощью оператора typedef. Благодаря данному оператору можем давать псевдонимы типам данных.
   Применение typedef в программе:
   ```
   typedef short int SHINT
   typedef unsigned long UNLONG
   ```

3. Число байт, необходимое для хранения enum без констант – 4 байта
   Число байт, необходимые для хранения enum с константами – 4 байта
   Размер enum совпадает с размером int, так как enum хранится в памяти компьютера как int

4. Объявим следующую структуру:
   ```
   Struct TypeData {
       int num;
       char ch;
       bool flag;
       float dub;
   } data;
   ```
   Размер этой структуры равен 12 байта, хотя сумма размеров полей 4 (int) + 1(char) + 1(bool) + 4(float) = 10 байт
   Это происходит из-за выравнивания адреса в памяти C++. Максимальный размер выравнивания 4 байта (int). Выравниваются в памяти поля по границе кратной своему же размеру.
   - 1-байтовые поля не выравниваются
   - 2-байтовые поля – выравниваются на чётные позиции
   - 4-байтовые – на позиции кратные четырём и т.д

   Сумма оставшихся полей равна 3 байтам, поэтому в данной ситуации компилятор вставляет ‘невидимые” байты для выравнивания. По этой причине, размер структуры равен 12, а не 10.

5. Компилятор C++ позволяет преобразовать все основные типы данных к этим же основным типам данных. Однако мы не можем преобразовывать пользовательские типы данных (struct, class) к основным:
   ```
   Jun = char(f); // Компилятор выдаст ошибку
   int b = 90;
   b = b / 3.6; // Потеря данных
   ```
   Исключения составляет тип данных enum, так как исходя из пункта 3 enum хранится как int:
   ```
   int f;
   f = char(data); // Компилятор допускает такое выражение
   data = char(f); // Компилятор выдаст ошибку
   ```
   Также из-за неправильного преобразования типов данных может возникать неожиданный результат:
   ```
   b = 90;
   b = (float)b / 3.6; // Может  возникнуть переполнение, не проверяется во время компиляции
   b = static_cast<int>(b / 3.6); // Оператор для явного преобразования типов
   ```

6. С помощью оператора typeid мы можем узнать тип объектов в режиме реального времени.

```
cout << “typeid TypeData: “ << typeid(data).name() << endl;
cout << “typeid double: “ << typeid(3.03).name() << endl;
```

Вывод в консоль:

```
TypeData
d
```

Текст программы:

```
#include <iostream>

struct TypeData {
   int num;
   char ch;
   bool flag;
   float dub;
} data;

int main() {
    int num;
    char ch;
    bool flag;
    float fl;
    double dub;
    wchar_t wh;

    std::cout << "****************** Exercise 1 ****************" << std::endl;
    std::cout << "int: " << sizeof(num) << " bytes" << std::endl;
    std::cout << "char: " << sizeof(ch) << " bytes" << std::endl;
    std::cout << "bool: " << sizeof(flag) << " bytes" << std::endl;
    std::cout << "float: " << sizeof(fl) << " bytes" << std::endl;
    std::cout << "double: " << sizeof(dub) << " bytes" << std::endl;
    std::cout << "wchar_t: " << sizeof(wh) << " bytes" << std::endl << std::endl;

   std::cout << "****************** Exercise 2 ****************" << std::endl;
   typedef short int SHINT;
   typedef unsigned long UNLONG;
   std::cout << "unsigned long" << " ----> " <<  typeid(UNLONG).name() << std::endl << std::endl;

   std::cout << "****************** Exercise 3 ****************" << std::endl;
   enum NumberMounths {JUNUARY, FEBRUARY, MARCH, APRIL};
   enum Lesson {FOUR = 4, FIVE = 5, SIX = 6, SEVEN = 7};
   NumberMounths Jun(JUNUARY), Feb(FEBRUARY), Mar(MARCH), Apr(APRIL);
   Lesson four(FOUR), five(FIVE), six(SIX), seven(SEVEN);
   int a = four, b = five, c = six, d = seven;
   std::cout << "NumberMounths: " << sizeof(NumberMounths) << " bytes" << std::endl;
   std::cout << "Lesson: " << sizeof(Lesson) << " bytes" << std::endl;
   std::cout << "Jun: " << Jun << '\n' << "Feb: " << Feb << '\n'
         << "Mar: " << Mar << '\n' << "Apr: " << Apr << std::endl;
   std::cout << "Enum: " << a << ", " << b << ", " << c << ", " << d << std::endl;
   std::cout << four << " -----> " << static_cast<char>(four) << std::endl; // Явное преобразование типа

   if (Jun == NumberMounths::JUNUARY) {
      std::cout << "The mounth is Junuary" << std::endl << std::endl;
   } else if (Jun == NumberMounths::FEBRUARY) {
      std::cout << "The mounth is February" << std::endl << std::endl;
```

```
    }

    std::cout << "****************** Exercise 4 ****************" << std::endl;
    std::cout << "TypeData: " << sizeof(data) << " bytes" << std::endl;
    std::cout << "BasicType: " << sizeof(num) + sizeof(ch) + sizeof(flag) + sizeof(fl) << " bytes" <<
std::endl;

    TypeData typeData;

    std::cout << "TypeData int: " << sizeof(typeData.num) << " bytes" << std::endl;
    std::cout << "TypeData char: " << sizeof(typeData.ch) << " bytes" << std::endl;
    std::cout << "TypeData bool: " << sizeof(typeData.flag) << " bytes" << std::endl;
    std::cout << "TypeData float: " << sizeof(typeData.dub) << " bytes" << std::endl << std::endl;

    std::cout << "****************** Exercise 5 ****************" << std::endl;
    {
        int a = 49;
        char ch = static_cast<char>(a);
        std::cout << a << " ----> " << ch << std::endl;
        int b = 90;
        b = b / 3.6; // Потеря данных
        b = 90;
        b = (float)b / 3.6; // Может возникнуть переполнение
        std::cout << b << std::endl;
        b = 90;
        std::cout << static_cast<int>(b / 3.6) << std::endl << std::endl;
    }
    int f;
    f = char(Jun);
    // Jun = char(f); // выдаст ошибку

    std::cout << "****************** Exercise 5 ****************" << std::endl;
    std::cout << "typeid TypeData: " << typeid(data).name() << std::endl;
    std::cout << "typeid duble: " << typeid(3.03).name() << std::endl;

    return 0;
}
```

Вывод: Я практическим путем выяснил какой размер занимают типы данных языка C++, пользовательские типы данных enum, struct и особенности распределения памяти компилятором.