Лабораторная работа № 4

Программирование низкоуровневых задач

Цель работы

1. Изучить представление различных типов и структур данных в памяти ЭВМ.
2. Освоить средства языка и стандартной библиотеки C++ для низкоуровневых манипуляций с битами данных, адресами памяти и строками C.

Задание на лабораторную работу

Общее задание

1. Подготовить инструменты для исследований и отладки.

   Написать функции для печати отдельных байт и блока данных `data` размером `size` байт в шестнадцатеричном и в двоичном представлении.

   ```
   void print_in_hex(uint8_t byte);
   void print_in_hex(const void* data, size_t size);
   void print_in_binary(uint8_t byte);
   void print_in_binary(const void* data, size_t size);
   ```

   Указание. Для удобства чтения рекомендуется между байтами добавлять пробелы и делать перевод строки, например, после каждых 16-и байт (в `print_in_hex()`) или каждых 4-х байт (в `print_in_binary()`).

2. Написать программу-калькулятор для побитовых операций.

   Пользователь вводит первый операнд, затем оператор (`&`, `|` или `^`), затем второй операнд. Программа выполняет указанное действие над операндами, и печатает расчет в шестнадцатеричном и двоичном виде. Операнды — двухбайтовые беззнаковые целые числа (`uint16_t`).

Пример ввода	**Соответствующий вывод**

`1025 & 127`	`01 04 & 7F 00 = 01 00`
	`00000001 00000100 &`
	`01111111 00000000 =`
	`00000001 00000000`

3. Изучить представление и размещение данных в памяти.

   3.1. Определить структуру `Student`, описывающую студента атрибутами:

   1) имя длиной не более 16 *печатных (printable)* символов;
   2) год поступления;
   3) средний балл;
   4) пол, представленный одним битом (0 — женский, 1 — мужской);
   5) количество пройденных курсов;

6) указатель на структуру `Student`, описывающую старосту группы (для старосты — нулевой указатель).

Указание. Поле размером в несколько бит (не больше, чем бит в определенном целочисленном типе) можно объявить так:

```
целочисленный-тип имя-поля : число-бит;
```

3.2. Объявить и заполнить массив из трех структур `Student`, описывающий двух студентов одной группы и их старосту.

3.3. Напечатать, занести в отчет и письменно пояснить:

1) адрес и размер массива;
2) адреса и размеры всех элементов массива;
3) для всех полей, кроме пола[1], одного из элементов массива (не старосты): адрес, смещение от начала структуры, размер, шестнадцатеричное и двоичное представление;
4) все элементы массива в шестнадцатеричном виде с указанием соответствия блоков байт полям структур.

Указание. Смещение поля `field` структуры *типа* `type` от начала любого её экземпляра можно определить макросом `offsetof(type, field)`.

4. Написать программу для обработки текстового файла, представляя текст только строками C, размещаемыми в динамической памяти.

4.1. Запросить у пользователя имя файла, сохранив его в массиве символов, размещенном на стеке (не в динамической памяти).

4.2. Проверить, используя функции стандартной библиотеки C++ для работы со строками C, что введенное имя файла корректно:

1) не содержит запрещенных символов: `*`, `"`, `<`, `>`, `?` или `|`;
2) если содержит двоеточие, то только вторым символом, которому предшествует буква, и за которым следует обратная косая черта (`\`).
3) расширение `*.txt` (в любом регистре).

Указание. Задачи решаются стандартными функциями `isalpha()`, `strchr()`, `strrchr()`, `strncmp()`.

4.3. Загрузить содержимое текстового файла в память целиком:

1) использовать `ifstream` или `fopen()` для доступа к файлу;

[1] К битовым полям нельзя применять оператор **`sizeof`** (так как он возвращает размер в байтах) и оператор взятия адреса (так как ячейки памяти менее байта не адресуются по определению).

2) использовать методы `seekg()` и `tellg()` либо функции `fseek()` и `ftell()` для определения размера файла, переместившись в его конец и получив текущее положение в файле;
3) выделить в динамической памяти массив достаточного размера;
4) загрузить всё содержимое файла в выделенную область памяти методом `read()` или функцией `fread()`.

4.4. Запросить у пользователя строку, поместив её в массив на стеке.

4.5. Найти введенную строку в загруженном тексте, не используя функций стандартной библиотеки C++ для работы со строками C.

4.6. Разбить текст файла на предложения, поместив их в динамический массив строк типа **`char`**`**` (указатель на массив указателей).

4.7. Упорядочить предложения по возрастанию их длины любым способом.

4.8. Записать все предложения в новый текстовый файл.

4.9. Освободить все выделенные в процессе решения блоки памяти.

Контрольные вопросы

1. Что такое режимы «жесткого» (hard) и «мягкого» (soft) реального времени? Какие особенности C++ делают его удобным для программ реального времени, а какие средства C++ недопустимы для таких программ?
2. Объясните подход «захват ресурса есть инициализация» (RAII) и какие языковые средства C++ в нем применяются. Какие ошибки RAII помогает предотвращать, и почему это важно в низкоуровневых задачах?
3. Опишите назначение и использование операторов **`new`** и **`delete`**. Чем отличается синтаксис удаления массива от синтаксиса удаления одиночного значения?
4. Опишите проблему фрагментации памяти и общие подходы к её решению.
5. Каковы особенности арифметики типов с плавающей запятой по сравнению с математически точной и сравнения их значений?
6. Как и почему корректно сравнивать значения с плавающей запятой и бороться с ошибками округления при операциях над такими значениями?
7. В чем заключается арифметика с фиксированной запятой, какие у нее преимущества и недостатки по сравнению с арифметикой типов с плавающей запятой?
8. Опишите действие оператора **`reinterpret_cast`**. В каких случаях его удобно применять, и какие проблемы при этом могут возникнуть?
9. Что такое выравнивание данных (alignment)? Почему оно существует, зачем и как контролировать его наличие?

10. Как определить размер переменной в C++, и в каких случаях он отличается от размера полезных данных, связанных с переменной?
11. Какие побитовые операторы имеются в C++? Приведите примеры их работы.
12. Что такое битовые флаги и битовые маски? Как в C++ записываются числа в системах счисления, отличных от десятичной?
13. Каким образом можно: а) объявить целочисленную переменную размером 8, 16, 32 бита (гарантированно); б) объявить битовый массив произвольно большого размера; в) поле структуры размера, не кратного байту (например, 9 бит)?
14. Как в C++ объявляются простые массивы (одно- и многомерные), каков синтаксис их инициализации, доступа к отдельным элементам, определения размера?
15. Опишите шаблон класса `std::array<T, N>`, его назначение и преимущества использования по сравнению с простыми массивами.
16. Что такое строки в стиле C (C-style string) и какие средства работы с ними имеются в стандартной библиотеке C++?