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.
3. Освоить поиск и чтение спецификаций реальных двоичных форматов данных с целью их разбора и формирования в прикладных программах.

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);
   ```

   Примечание. Нетипизированный указатель **`void`**`*` может содержать адрес любых данных, однако перед разыменованием должен быть приведен к типизированному указатели (в данном случае удобно к `uint8_t*`).

   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`

   Примечание. Байты целых чисел выводятся в порядке от младшего к старшему ($1025_{10} = 0401_{16}$, а не 0104_{16}) — это нормальный результат.

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 определенным 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 `|`;

[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) 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;
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.

Задания по вариантам

Стеганография

Стеганография — это сокрытие одних данных в других. Например, текст может быть скрыт в изображении так, что изображение заметно человеку не изменится, но из небольших искажений, внесенных при скрытии, текст сможет быть восстановлен. Необходимо написать программу, которая скрывает введенный пользователем текст в указанном изображении или восстанавливает из указанного изображения текст.

Любой цвет может быть представлен сочетанием красного, зеленого и синего цветов (red, green, blue — RGB). Например, цвета вида (*x*, *x*, 0) задают оттенки желтого цвета как смеси красного и зеленого цветов в равной пропорции. Каждой точке изображения соответствует свой цвет.

Если на кодирование цвета отведено 24 бита (3 байта), целесообразно кодировать каждую компоненту одним байтом со значениями от 0 до 255. Изменение младшего бита любой компоненты изменит её незначительно — на единицу, то есть примерно на 0,5 %. Разница в итоговом цвете не будет заметна человеку, поэтому в целях стеганографии можно заменять младшие биты компонент цвета на биты сообщения. Собрав сообщение из тех же бит, можно восстановить его (если оно ранее было скрыто в изображении). Рисунок 1 иллюстрирует предложенный метод.

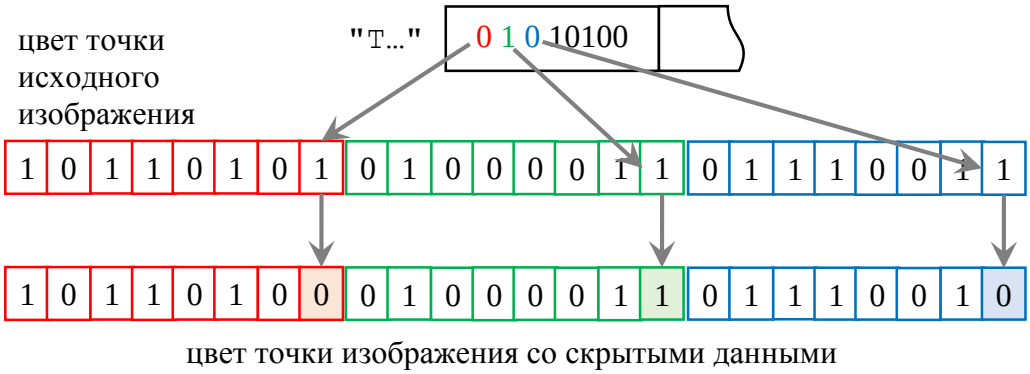

Рисунок 1 — Процес стеганографического сокрытия сообщения

Вариант 1

Изображение в формате BMP (вариант Windows NT и выше) без палитры в 24-битном цвете без сжатия RLE. Допустимо воспользоваться заголовочным файлом `<windows.h>`, в котором определены структуры заголовков `BITMAPFILEHEADER` и `BITMAPINFOHEADER`.

Вариант 2

Изображение в формате Targa (TGA) без палитры в 24-битном цвете без сжатия RLE. Просмотр файлов TGA возможен бесплатной программой IrfanView.

Редактирование изображений

Любой цвет может быть представлен сочетанием красного, зеленого и синего цветов (red, green, blue — RGB). Например, цвета вида (*x*, *x*, 0) задают оттенки желтого цвета как смеси красного и зеленого цветов в равной пропорции. Каждой точке изображения соответствует свой цвет.

Если в изображении немного цветов (например, это график), рационально не записывать цвет каждой точки тремя компонентами, а составить таблицу (палитру) всех цветов изображения, а для точки указывать индекс цвета в палитре. В простейшем случае, когда в палитре два цвета, битом «0» кодируется точка одного цвета, а битом «1» — точка другого цвета.

Если на кодирование цвета отведено 16 бит, и палитра не используется, нужно разделить 16 бит на три компоненты. Поровну это сделать нельзя, и на зеленый цвет отводится не 5, а 6 бит. Таким образом, ни одна компонента не занимает байт целиком — нужно использовать битовые маски и сдвиги, чтобы получить значения компонент.

Необходимо написать программу, создающую новое изображение в соответствии с вариантом задания.

Вариант 3

Перевод изображения в формате BMP с палитрой в 16-битном цвете (5-6-5 бит) в оттенки серого. Для получения соответствующего оттенка нужно заменить все компоненты цвета на их гармоническое среднее.

Вариант 4

Изображение графика функции $y(t) = 1 - \sin(t) \cdot e^{-t}$ в формате монохромного BMP (1 бит на точку) размером 640×480 точек. Масштаб следует выбрать такой, чтобы график был хорошо различим.

Работа с архивом tar (вариант 5)

Необходимо написать программу или программы для создания и распаковки архивов без сжатия формата tar в ограниченном объеме:

1) обрабатывать только простые файлы в рабочем каталоге программы (или в корне архива);
2) не записывать в архив и не восстанавливать имя владельца файла (owner), группу файла (group), время последнего изменения (last modification time), а также любые поля, не нужные для п. п. 1) и 3).
3) записывать и проверять контрольные суммы заголовков файлов в архиве.

Работать с форматом tar может большая часть архиваторов, например, 7zip. Правильно сформированный архив должен успешно открываться этими программами.

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 проблему фрагментации памяти и общие подходы к её решению. Какие средства C++ и стандартной библиотеки позволяют бороться с фрагментацией?
4. Опишите действие оператора **`reinterpret_cast`**. В каких случаях его удобно применять, и какие проблемы при этом могут возникнуть?
5. Что такое выравнивание данных (alignment)? Почему оно существует, зачем и как контролировать его наличие?
6. Как определить размер переменной в C++, и в каких случаях он отличается от размера полезных данных, связанных с переменной?
7. Какие побитовые операторы имеются в C++? Приведите примеры их работы.
8. Что такое битовые флаги и битовые маски? Как в C++ записываются числа в системах счисления, отличных от десятичной?
9. 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)?
10. Как в C++ объявляются простые массивы (одно- и многомерные), каков синтаксис их инициализации, доступа к отдельным элементам, определения размера?
11. Опишите шаблон класса `std::array<T, N>`, его назначение и преимущества использования по сравнению с простыми массивами.
12. Что такое строки в стиле C (C-style string) и какие средства работы с ними имеются в стандартной библиотеке C++?
13. Опишите назначение и использование операторов **`new`** и **`delete`**. Чем отличается синтаксис удаления массива от синтаксиса удаления простой переменной?