Лабораторная работа №2
Тема: Объединения, поля бит.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1. Объединения

Объединения СИ очень похожи на структуры, за исключением того, как СИ хранит их в памяти; кроме того, объединение может хранить значение только для одного элемента в каждый момент времени.

Объединение представляет собой структуру данных, подобную структуре СИ, и состоит из частей, называемых элементами.

Объединение определяет шаблон, с помощью которого программы далее объявляют переменные.

Для обращения к определенному элементу объединения ваши программы используют оператор СИ точку.

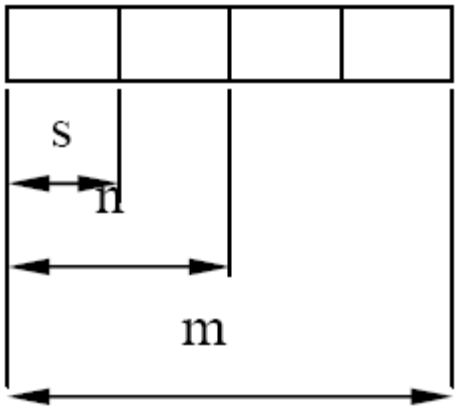
Чтобы изменить значения элемента объединения внутри функции, ваша программа должна передать переменную объединения в функцию с помощью адреса.

Анонимное объединение представляет собой объединение, у которого нет имени (тэга).

Как вы узнаете, объединения очень похожи на структуры Си, однако способ, с помощью которого Си хранит объединения, отличается от способа, с помощью которого Си хранит структуры.

Объявление объединения такое же самое, как и объявление структуры, только вместо специального слова struct используется union.

```
union example
{
  int n;
  float m;
  char s;
} var;
```



Выделение памяти под обьединение

По сути, объединения – это структуры с нулевым смещением элементов структуры относительно ее начала. Объем выделяемой памяти под объединение равен размеру самого большого элемента в объединении.

Разрешенные операции:

- можно присваивать объединения друг другу
- адрес брать так же ни кто не запрещал
- к элементам можно получить доступ, так же как и в структурах, т.е. через (.) или (->)

Внутри ваших программ объединения СИ очень похожи на структуры. Например, следующая структура определяет объединение с именем *distance*, содержащее два элемента:

```
union distance
{
  int miles;
  long meters;
};
```

Как и в случае со структурой, описание объединения не распределяет память. Вместо этого описание предоставляет шаблон для будущего объявления переменных.

Как и для структуры, ваша программа может присвоить значение любому элементу. Однако в отличие от структуры значение может быть присвоено только одному элементу в каждый момент времени. Когда вы объявляете объединение, компилятор СИ распределяет память для хранения самого большого элемента объединения. В случае объединения *distance* компилятор распределяет достаточно памяти для хранения значения типа *long*.

Предположим, что ваша программа присваивает значение элементу *miles*, как показано ниже:

```
japan.miles = 12123;
```

Если далее ваша программа присваивает значение элементу meters, значение, присвоенное элементу *miles*, теряется.

Следующая программа иллюстрирует использование объединения *distance*. Сначала программа присваивает значение элементу *miles* и выводит это значение. Затем программа присваивает значение элементу *meters*. При этом значение элемента *miles* теряется:

```
#include <iostream.h>
union distance
{
  int miles;
  long meters;
} walk;
```

```
     void main(void)
     {
        walk.miles = 5;
        cout<<"Пройденное расстояние в милях
"<<walk.miles<<endl;
        walk.meters = 10000;
        cout<<"Пройденное расстояние в метрах
"<<walk.meters<<endl;
     }
```

Как видите, программа обращается к элементам объединения с помощью точки, аналогичная запись использовалась при обращении к элементам структуры. Объединение хранит значение только одного элемента в каждый момент времени Объединение представляет собой структуру данных, которая, подобно структуре СИ, позволяет вашим программам хранить связанные части информации внутри одной переменной. Однако в отличие от структуры объединение хранит значение только одного элемента в каждый момент времени. Другими словами, когда вы присваиваете значение элементу объединения, вы перезаписываете любое предыдущее присваивание.

Объединение определяет шаблон, с помощью которого ваши программы могут позднее объявлять переменные. Когда компилятор СИ встречает определение объединения, он распределяет количество памяти, достаточное для хранения только самого большого элемента объединения.

2. Анонимные объединения

Анонимное объединение представляет собой объединение, у которого нет имени. СИ предоставляет анонимные объединения, чтобы упростить использование элементов объединений, предназначенных для экономии памяти или создания псевдонимов для определенного значения. Например, предположим, что вашей программе требуются две переменные *miles* и meters. Кроме того, предположим, что программа использует только одну из них каждый данный момент времени. В этом случае программа могла бы использовать элементы объединения, подобного уже обсуждавшемуся объединению *distance*, а именно *name.miles* и *name.meters*. Следующий оператор создает анонимное (безымянное) объединение:

```
union
{
  int miles;
  long meters;
};
```

Как видите, объявление не использует имя объединения и не объявляет переменную объединения. Программа, в свою очередь, может обращаться к элементам с именами *miles* и *meters* без помощи точки. Следующая

программа создает анонимное объединение, которое содержит элементы *miles* и *meters*. Обратите внимание, что программа трактует элементы как обычные переменные. Однако различие между элементами и обычными переменными заключается в том, что, когда вы присваиваете значение любому из этих элементов, значение другого элемента теряется:

```
#include <iostream.h>

union
{
  int miles;
  long meters;
};

void main(void)
{
   miles = 10000;
   cout << "Значение в милях " << miles << endl;
   meters = 150000;
   cout << "Значение в метрах " << meters << endl;
}
```

Как видите, с помощью анонимного объединения, программа может сэкономить память, не используя имя объединения и точку для обращения к значениям элементов.

✓ Анонимные объединения позволяют вашим программам экономить пространство

✓ Анонимное объединение представляет собой безымянное объединение.

✓ Анонимные объединения обеспечивают вашим программам способ экономии памяти, и при этом можно не использовать имя объединения и точку.

Следующие операторы определяют анонимное объединение, способное хранить две символьные строки:

```
union
{
   char short_name[13];
   char long_name[255];
};
```

3. Поля бит

Поле представляет собой последовательность соседних двоичных разрядов (бит) внутри одного целого значения. Каждое поле может иметь тип

unsigned int или *signed int* и размещается в машинном слове целиком, а вся группа полей может выходить за пределы машинного слова.

Поле бит – особый тип структуры, определяющий какую длину имеет каждый ее элемент. Общий вид объявления полей бит имеет вид:

```
struct имя структуры
{ тип имя 1: длина;
. . .
  тип имя n : длина;
};
```

Наиболее распространенный подход к использованию полей можно показать на примере объявления следующей структуры:

```
struct example
{
  int p1:1;
  unsigned p2:2;
  int:6;
  int p3:4
} p1;
```

Объявление такого вида обеспечивает размещение структуры (полей бит) в памяти следующим образом. В полях типа *signed* крайний левый бит является знаковым. Таким образом, поле *signed int p1:1* может быть использовано для хранения значений -1 и 0, так как любое не нулевое значение поля *p* будет интерпретироваться как -1. Поле un*signed int p2:2*, в отличие от *int p1:1*, может принимать 4 значения: 0, 1, 2 и 3. В объявлении структуры имеется еще два поля (*int:6*; int *p3:4*). Первое указывает, что структура содержит 6 неиспользуемых бит, второе предназначено для чисел в диапазоне от –7 до 7.

В Borland C самый левый бит является знаковым. Поля могут не иметь имени; с помощью безымянного поля (задаваемого только двоеточием и шириной) организуется пропуск требуемого количества разрядов. Ширина равная нулю, используется тогда когда необходимо выйти на границу следующего слова.

Все особенности, связанные с использованием полей, например: может ли поле байт перейти границу слова, зависят от аппаратной реализации.

При определенном удобстве работа с полями может повлечь некоторые трудности. Они связаны, например, с тем, что на одних машинах поля размещаются слева направо, на других - справа налево. Это в свою очередь влечет некоторые трудности при перенесении программ. Поля не могут быть массивами и не имеют адресов и, следовательно, операция & к ним не применима.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Варианты заданий

Задача 1 - максимум 4-5 баллов при сдаче на том же занятии, на котором задача выдана. При сдаче на следующих занятиях максимум - 2 балла.

Задача 2 - максимум 5 баллов при сдаче на том же занятии, на котором задача выдана. При сдаче на следующих занятиях максимум - 3-4 балла. Принимается частичное решение, если это позволяет условие задачи.

Задача 1 должна быть сдана обязательно.

Вариант	Задание
1	1. Реализовать структуру, представляющую адрес. Структура содержит перечисление, которое определяет тип задания адреса (минимум 3 типа), почтовый индекс и объединение структур, представляющих адрес определенного типа. Примеры адресов разного типа: местный городской адрес (город, улица, дом, квартира), местный сельский адрес (область, район, село, улица, дом), адрес другой страны (страна, произвольная строка детального адреса). Сделать возможность ввода адреса, вывода всех адресов определенного типа. 2. Добавить в городской адрес поля бит: этаж (максимальный - 200-й), корпус (максимальный - 30-й), подъезд (максимальный - 100-й). Добавить поиск адреса по его частичному указанию (например по запросу “Бров 6” должен выдаваться адрес “ул. Петруся Бровки 6”). Добавить в адрес координату (в декартовой системе координат - x и y), добавить возможность задания исходной координаты и радиуса. Вывести адреса в заданном радиусе вокруг заданной координаты.

2	1. Реализовать структуру, представляющую геометрическую фигуру. Структура содержит перечисление, которое определяет тип фигуры (минимум три типа) и объединение структур, представляющих основные параметры фигуры (сторона для квадрата, две стороны для прямоугольника, радиус для окружности и т.д.). Вывести площадь и периметр фигур. 2. Добавить в фигуру поля бит, содержащие информацию о том, включена ли обводка, цвет обводки (один из 15-ти), включена ли заливка, цвет заливки (один из 15-ти). Вывести все фигуры, у которых включена обводка, заливка, цвет обводки не совпадает с цветом заливки. Вывести площадь прямоугольника, в который вписана фигура.
3	1. Реализовать структуру, представляющую файл. Структура содержит имя, логин владельца, перечисление, которое определяет подтип: файл или директория. Также содержит объединение структур, представляющих файл или директорию. Структура, представляющая файловый подтип, содержит расширение и размер. Структура, представляющая подтип-директорию, содержит количество файлов, содержащихся внутри. Сделать возможность с клавиатуры добавлять файлы и директории, выводить все текущие файлы и директории. 2. Добавить в подтип “файл” поля бит, представляющее права доступа: бит доступа на чтение, бит доступа на запись и бит доступа на исполнение. Сделать возможность выполнять с клавиатуры (в ходе выполнения программы, не из командной строки) команду chmod, которая принимает имя файла и меняет права доступа. Добавить эмуляцию чтения, записи и исполнения файла, реакцию, соответствующую правам доступа (вывод сообщения “разрешено/запрещено/файл отсутствует”). Добавить в подтип “директория” массив файлов, хранящихся в директории, возможность добавления файлов и директорий в директорию.

4	1. Реализовать структуру, представляющую элемент пользовательского интерфейса на экране настроек приложения. Структура содержит перечисление, отображающее тип настройки (переключатель, текстовое поле, выбор из множества, слайдер) и объединение, хранящее текущее значение настройки (0/1 для переключателя, строку для текстового поля и выбора, число с плавающей запятой - для слайдера). Сделать возможность задать с клавиатуры настройки, изменить значение любого пункта, вывести значения всех пунктов. 2. Для текстового поля добавить типы (логин, пароль, телефон) и валидацию введенного значения. Добавить ко всем элементам поля бит: в фокусе элемент или нет, видим или нет, активен или нет, прозрачность от 0 до 100. Добавить к элементам координату левого верхнего угла и размеры, положение по оси Z. Для введенной точки с клавиатуры для выбранного элемента определить, видна ли эта точка с учетом положения по оси Z, флага видимости и значения прозрачности.
5	1. Реализовать структуру, представляющую цвет. Структура содержит тип цветовой модели (RGB, HSV, CMYK) и объединение, хранящее соответствующие структуры с необходимыми значениями компонент с плавающей точкой (например, красный, зеленый и синий для RGB). Добавить возможность задать цвет и перевести из любой модели в любую. 2. При помощи полей бит добавить возможность упаковать каждую из моделей в два байта (с потерей диапазона цветов). Выводить разницу между оригинальным и упакованным значением. Добавить в структуру “цвет” значение alpha - прозрачность. Вывести результирующий цвет при наложении одного цвета на другой. Добавить выбор из нескольких алгоритмов смешивания.

6	1. Реализовать структуру, представляющую вложение к письму. Структура содержит тип вложения (текстовый документ, изображение) и объединение структур, представляющих каждый из типов вложения. Текстовый документ содержит кол-во страниц и первый абзац текста. Изображение - размер в пикселях и URL. Сделать возможность составить письмо с приложением, проверить, валидность приложения к письму. Письмо невалидно, если в текстовом документе приложения больше 10000 страниц или протокол в URL изображения - http, а не https. 2. Добавить новый тип вложения - бинарный файл. Структура, представляющая бинарный файл содержит имя и поля бит: проверен ли на вирусы, больше ли максимального размера, из доверенного ли источника, исполняемый ли. Письмо невалидно, если бинарный файл не проверен на вирусы, больше максимального размера или не из доверенного источника. Добавить поиск по частичному совпадению текста во вложениях: внутри первого абзаца текста, в URL изображения, имени бинарного файла.
7	1. Реализовать структуру, представляющую расстояние. Структура содержит тип системы мер (метрическая, английская, старорусская) и объединение структур, представляющих каждый тип (хранит кол-во километров, метров и сантиметров для метрической системы, кол-во миль, футов и дюймов для английской системы и кол-во верст, саженей и пядей для старорусской системы). Сделать возможность ввода расстояния в каждой из систем и возможность перевода введенного расстояния из одной системы в другую. 2. При помощи полей бит использовать фиксированное кол-во бит для хранения метров и сантиметров, футов и дюймов, саженей и пядей (например, в сантиметрах нет смысла хранить значения более 100, т.к. 1 метр и 101 сантиметр - это 2 метра 1 сантиметр). Добавить возможность к текущему расстоянию добавить любое значение в любой системе мер и увидеть при этом результат.

8	1. Реализовать структуру, представляющую небесное тело. Структура содержит тип тела (звезда, планета, астероид), его имя, массу, расстояние от земли и объединение структур, представляющих соответствующий тип тела. Звезда содержит массив указателей на планеты, планета - массив основных элементов в составе и массив имен спутников, астероид - скорость. В коде задать солнечную систему. Сделать возможность просмотра всех объектов солнечной системы. 2. В структуру, представляющую планету добавить среднюю температуру, поля бит: пригодна ли планета для жизни, порядковый номер по расстоянию от звезды (максимальный - 1000), посещена ли человеком. Сделать возможность добавления объектов. Добавить возможность с клавиатуры выбрать две звезды, которые программно сталкиваются вместе с планетами, в результате чего генерируются новые объекты.
9	1. Объявить два перечисления: арабские цифры (0-9) и римские цифры (I, V, X, L, C, D, M). Реализовать структуру, представляющую число. Число может быть записано одним из двух наборов цифр. Структура хранит тип цифр (римские, арабские) и объединение массивов, хранящих каждый из типов цифр. Префиксную запись римских чисел можно проигнорировать, т.е. можно считать, что запись “IV” для обозначения 4 в данном случае невалидна, а 4 записывается как “IIII”, в результате чего перевод заключается исключительно в сложении цифр. Реализовать ввод чисел каждого типа, вывод введенного числа, записанного другими цифрами. 2. Добавить возможность сложения двух чисел, записанных разными цифрами. Добавить возможность упаковки (уменьшения используемого кол-ва памяти) нескольких цифр при помощи использования полей бит.

10	1. Реализовать структуру, представляющую ресурс. Структура содержит идентификатор, тип хранения ресурса (оперативная память, диск, сеть) и объединение структур, представляющих соответствующий тип: память содержит целочисленный адрес, диск - путь к файлу, сеть - url. Сделать возможность добавления сетевых ресурсов с клавиатуры. Добавить эмуляцию загрузки ресурса с определенным идентификатором из сети по запросу с клавиатуры: ресурс появляется в памяти и на диске. Сделать возможность вывода всех текущих ресурсов. 2. Добавить эмуляцию выхода из приложения. При этом все ресурсы, хранящиеся в памяти очищаются. Добавить эмуляцию очистки дискового кэша. При этом все дисковые ресурсы очищаются. К каждому ресурсу добавить поле бит, обозначающее факт обращения к ресурсу: при обращении флаг выставляется. Добавить возможность сбросить флаг обращения у всех ресурсов. Реализовать логику, при которой, если ресурс присутствует в памяти, то обращения к диску и сети не происходит, если ресурс находится на диске, то обращения к сети не происходит.