ЛАБОРАТОРНАЯ РАБОТА № 12
ИСПОЛЬЗОВАНИЕ СТРУКТУР В ПРОГРАММАХ

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Структура – это создаваемый программистом новый тип данных. Структура представляет собой набор из одной или нескольких переменных, возможно даже разных типов, *объединенных вместе под одним именем*.

Переменные, которые объединены структурой, называются **полями** и могут иметь любой тип, кроме типа этой же структуры, но могут быть указателями на него.

В языке *C* синтаксис объявления структуры следующий:

```
struct <имя_типа>
{
  <тип_1> <элемент_1>;
  <тип_2> <элемент_2>;
  . . . . . .
  <тип n> <элемент n>;
};
```

Объявляется структура обычно в заголовочном файле или в файле с текстом программы, но до описания функций, в которых они используются. При этом создается *новый тип переменных* и появляется возможность использовать в программе переменные этого нового типа.

Структуры удобно применять тогда, когда в программе используются понятия, характеризующиеся несколькими параметрами, например, дата (включает день, месяц и год), время (час, минута и секунда), комплексное число (реальная и мнимая части), геометрическая фигура, информация о человеке и т.д.

Например, для описания даты необходимо три переменных, в которых хранится день, месяц и год. Объединить их вместе удобно с помощью следующей структуры:

```
struct Date       // структура для хранения даты
{
  int Day;      // поле для хранения дня
  int Month;    // поле для хранения месяца
  int Year;     // поле для хранения года
};
```

Поместив в программу данное объявление, мы создаем новый тип переменных – `Date`, предназначенный для хранения даты. Его размерность будет 12 байт. Чтобы создать переменную этого типа нужно просто объявить ее в нужном месте программы:

```
Date Birthday;   // объявление переменной Birthday типа Date
Date d1, d2, d3; // объявление трех переменных типа Date
Date Arr[10];    // объявление массива переменных типа Date
```

Имя типа при объявлении структуры может отсутствовать, в этом случае должен быть указан список переменных, указателей или массивов.

```
struct
{
  unsigned char Day;
  unsigned char Month;
  unsigned short Year;
} Birthday, d1, d2, d3, Arr[10];
```

Размер структуры Date можно сократить, если использовать другие типы полей. Действительно, для хранения дня и месяца достаточно по одному байту, а для года – два байта. Кроме того, эти параметры всегда положительные:

```
struct Date  // Новый вариант структуры для хранения даты
{
  unsigned char Day;      // поле для хранения дня
  unsigned char Month;    // поле для хранения месяца
  unsigned short Year;     // поле для хранения года
};
```

Еще примеры структур:

```
// структура для хранения параметров точки на плоскости
struct Point
{
   int x;
   int y;
};
// структура для хранения параметров прямоугольника
struct Rect
{
   int left;  // Координаты левого
   int top;   //                  верхнего угла
   int right;  // Координаты правого
   int bottom; //                  нижнего угла
};
```

Для создания переменных этих типов объявляем в нужном месте программы:

```
Point  a, b;
Point  ArrayPoints[100];  // Массив точек
Rect r1, r2;
```

Для работы с комплексными числами удобно использовать структуру, поля которой можно сделать типа **float** или **double**:

```
struct Complex  // структура для хранения комплексного числа
{
   float re;  // реальная часть
   float im;  // мнимая часть
};
```

Полями структуры могут быть объекты других структур, например, в структуре, предназначенной для хранения информации о студенте, поле «*День рождения*» удобно реализовать с помощью объявленной ранее структуры **Date**:

```
// структура для хранения данных студента
struct Student
{
  char Fio[30]; // Фамилия, имя, отчество – строка из 30 символов
  Date Birth;   // Дата рождения – используется структура Date
  char Course;  // Курс
  char Group;   // Группа
  float Gpa;    // Grade Point Average – средний балл
};
```

В структуре, предназначенной для хранения параметров прямоугольника, можно использовать структуры **Point** (точка):

```
struct Rectangle   // прямоугольник (2 точки)
{
  Point lt;   // координаты верхнего левого угла
  Point rb;   // координаты нижнего правого угла
};
```

При объявлении переменной типа какой-нибудь структуры можно сразу провести ее **начальную инициализацию**. Для этого значения ее элементов перечисляют в фигурных скобках в порядке их описания:

```
Date d = {18, 9, 2011};
Student s1 = {"Иванов А.Б.", {18, 3, 2002}, '1', '5', 4.2};
```

Для обращения ко всей структурной переменной используется ее *имя*, **доступ к полям структуры** выполняется с помощью оператор выбора **'.'** (точка), например:

```
Date d;       // объявление переменной d типа Date
d.Day = 29;  // присваивание полю Day переменной d значения 29
d.Month = 2;     // присваивание полю Month значения 2
d.Year = 2012;  // присваивание полю Year значения 2012
int k = d.Day; // Значение поля Day переменной d присваивается k
```

В программе можно объявлять указатели на структуры. **При обращении к полю структуры через указатель** используется оператор **'->'**, например:

```
Date *pd;
pd = new Date; // Динамическое создание структурной переменной
pd->Day = 29;
pd->Month = 2;
pd->Year = 2012;
int k = pd->Day;
```

С полем структуры можно работать так же, как и с переменной соответствующего типа: числовые переменные могут участвовать в арифметических выражениях, со строками можно выполнять все стандартные операции.

```
Date d = {18, 9, 2011};
d.Day++;

Point p1 = {10, 20};
Point p2 = {21, 35};
p1.x++; // Сдвиг точки по оси x
p1.y++; // Сдвиг точки по оси y
int k = p2.x - p1.x;
```

Для переменных одного и того же структурного типа определена операция присваивания, при этом происходит поэлементное копирование.

```
Date d = {18, 9, 1991};
Student St;
St.Birth = d;
```

Структуру можно **передавать в функции в качестве параметра**. Например, напишем функцию, которая выводит на дисплей дату в формате: **25.12.2020**.

```
void PrintDate(Date dat)
{
  printf("%02d.%02d.%04d\n", dat.Day, dat.Month, dat.Year);
}
```

Теперь, используя эту функцию можно выводить любые даты на дисплей:

```
Date d = {18, 9, 2011};
PrintDate(d);  // Выводит на дисплей 18.09.2011
d.Day++;
PrintDate(d);  // Выводит на дисплей 19.09.2011
d.Day++;
PrintDate(d);  // Выводит на дисплей 20.09.2011
```

Структуру можно использовать **в качестве возвращаемого значения функции**. Например, напишем функцию, которая формирует дату из отдельных переменных:

```
Date SetDate(unsigned char d, unsigned char m, unsigned short y)
{   // d - день, m - месяц, y - год
  Date tmp;    // Создаем временную локальную переменную функции
  tmp.Day = d;
  tmp.Month = m;
  tmp.Year = y;
  return tmp;  // Возвращаем tmp
}
```

Пример использования функций **SetDate** и **PrintDate:**

```
int main()
{
  unsigned short a, b, c;
  Date d1;

  d1 = SetDate(30, 11, 2020);
  PrintDate(d1);

  printf("\n Введите дату\n");
  printf(" день: ");
  scanf("%d", &a);
  printf(" месяц: ");
  scanf("%d", &b);
  printf(" год: ");
  scanf("%d", &c);
  d1 = SetDate(a, b, c);

  printf("\nВы ввели дату:\n");
  PrintDate(d1);
}
```

Для работы с комплексными числами удобно не только использовать структуру **Complex**, но и функции для работы с ними. Например, функция суммирования двух комплексных чисел должна принимать два аргумента типа **Complex** и возвращать тоже переменную типа **Complex**:

```
Complex Summ(Complex a, Complex b)
{
  Complex tmp;

  tmp.re = a.re + b.re;
  tmp.im = a.im + b.im;
  return tmp;
}
```

Функция вычисления модуля комплексного числа возвращает уже не комплексное число, а вещественное, в нашем примере типа **float**:

```
float Mod(Complex  a)
{
  float f;
  f = sqrt(a.re * a.re + a.im * a.im);
  return f;
}
```

Для вывода на дисплей тоже удобно написать функцию:

```
void PrintComplex(Complex a)
{
  printf("%f + %f * j\n", a.re, a.im);
}
```

Но данная функция не всегда корректно выводит информацию, например, когда мнимая часть меньше нуля, то нужно выводить знак минус, а не плюс, а когда мнимая часть равна нулю, то ее вообще выводить не надо. Поэтому перепишем функцию так:

```
void PrintComplex(Complex a)
{
  if (a.im > 0)
    printf("%f + %f * j\n", a.re, a.im);
  else if (a.im < 0)
    printf("%f - %f * j\n", a.re, -a.im);
  else
    printf("%f\n", a.re);
}
```

Используя эти и подобные функции можно написать программу, работающую с комплексными числами:

```
int main()
{
  Complex  c1, c2, c3;
  float m;

  printf("\n Введите первое число\n");
  printf(" re: ");
  scanf("%f", &c1.re);
  printf(" im: ");
  scanf("%f", &c1.im);
  printf("\n Введите второе число\n");
  printf(" re: ");
  scanf("%f", &c2.re);
  printf(" im: ");
  scanf("%f", &c2.im);

  printf("\n Первое число: ");
  PrintComplex(c1);
  printf(" Второе число: ");
  PrintComplex(c2);
  c3 = Summ(c1, c2);
  printf(" Сумма: ");
  PrintComplex(c3);
  m = Mod(c3);
  printf(" Модуль: %f\n", m);
}
```

Битовые поля — это особый вид полей структуры. Они используются для плотной упаковки данных. При описании битового поля после имени через двоеточие указывается длина поля в битах (целая положительная константа):

```
struct BDate
{
  unsigned char Day : 5;       // Занимает 5 бит
  unsigned char Month : 4;     // Занимает 4 бит
  unsigned short Year : 7;     // Занимает 7 бит
};
```

Битовые поля могут быть любого целого типа.. Доступ к полю осуществляется обычным способом — по имени.

```
BDate bd;
bd.Day = 16;
bd.Month = 12;
bd.Year = 20;
```

Адрес поля получить нельзя, однако в остальном битовые поля можно использовать точно так же, как обычные поля структуры. Следует учитывать, что операции с отдельными битами реализуются гораздо менее эффективно, чем с байтами и словами, так как компилятор должен генерировать специальные коды, и экономия памяти под переменные оборачивается увеличением объема кода программы. Размещение битовых полей в памяти зависит от компилятора и аппаратуры.

Объединение (`union`) представляет собой частный случай структуры, все поля которой располагаются по одному и тому же адресу. Формат описания такой же, как у структуры, только вместо ключевого слова `struct` используется слово `union`. Длина объединения равна наибольшей из длин его полей, В каждый момент времени в переменной типа объединение хранится только одно значение, и ответственность за его правильное использование лежит на программисте. Объединения применяют для экономии памяти в тех случаях, когда известно, что больше одного поля одновременно не требуется или когда нужно обеспечить доступ к одним и тем же байтам разными способами:

```
union FloatChar     // Объединение - к одним и тем же байтам
{                         // можно обращаться как
  float v;                // к переменной типа float
  unsigned char b[4];     // или как к массиву из 4 байт
};
```

Используя данное объединение можно написать программу, которая позволяет просматривать побайтное представление чисел типа **float**.

```
int main()
{
  FloatChar pf;

  printf("Float = ");
  scanf("%f", &pf.v);

  printf("%f\n", pf.v);
  printf("%02X %02X %02X %02X\n",pf.b[3],pf.b[2],pf.b[1],pf.b[0]);
  return 0;
}
```

ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

В начале программы ОБЯЗАТЕЛЬНО выводить:
ФИО, группа, номер лаб. работы, номер варианта.

Задание. Разработать структуры, функции, использующие эти структуры, и программу, демонстрирующую работу данных функций. Вариант задания выдает преподаватель.

Для получения максимального балла добавить в программу меню и возможность проводить повторные вычисления.

1. Разработать структуру «Прямоугольник», прямоугольник задается координатами его верхнего левого и правого нижнего углов. Написать функцию, которая вычисляет и возвращает длину диагонали прямоугольника. Написать функцию, которая вычисляет и возвращает площадь прямоугольника. Написать функцию, которая вычисляет и возвращает периметр прямоугольника. Использовать данную структуру. Написать функции ввода данных прямоугольника и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

2. Разработать структуру «Равносторонний треугольник», треугольник задается координатами его центра и длиной стороны. Написать функцию, которая вычисляет и возвращает площадь треугольника. Написать функцию, которая вычисляет и возвращает периметр треугольника. Использовать данную структуру. Написать функции ввода данных треугольника и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

3. Разработать структуру «Квадрат», квадрат задается координатами его центра и длиной стороны. Написать функцию, которая вычисляет и возвращает длину диагонали квадрата. Написать функцию, которая вычисляет и возвращает площадь квадрата. Написать функцию, которая вычисляет и возвращает площадь описанной вокруг квадрата окружности. Использовать данную структуру. Написать функции ввода данных квадрата и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

4. Разработать структуру «Окружность», окружность задается координатами ее центра и радиусом. Написать функцию, которая вычисляет и возвращает площадь круга. Написать функцию, которая вычисляет и возвращает длину окружности. Написать функцию, которая вычисляет и возвращает длину стороны квадрата, вписанного в окружность. Использовать данную структуру. Написать функции ввода данных круга и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

5. Разработать структуру «Прямоугольник», прямоугольник задается координатами его верхнего левого и правого нижнего углов. Написать функцию, которая сравнивает два прямоугольника по площади и возвращает тот прямоугольник, площадь которого больше. Написать функцию, которая вычисляет и возвращает разность площадей двух прямоугольников. Использовать данную структуру. Написать функции ввода данных прямоугольника и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

6. Разработать структуру «Квадрат», квадрат задается координатами его центра и длиной стороны. Написать функцию, которая сравнивает два квадрата по площади и возвращает площадь того, который больше. Написать функцию, которая определяет, находится ли начало координат внутри квадрата . Использовать данную

структуру. Написать функции ввода данных квадрата и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

7. Разработать структуру «Окружность», окружность задается координатами ее центра и радиусом. Написать функцию, которая сравнивает две окружности по площади и возвращает площадь той, которая меньше. Написать функцию, которая определяет, центр какой окружности находится ближе к началу координат, и возвращает ее. Использовать данную структуру. Написать функции ввода данных окружности и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

8. Разработать структуру «Отрезок», отрезок задается координатами на плоскости его начала и конца. Написать функцию, которая вычисляет и возвращает длину отрезка. Написать функцию, которая сравнивает два отрезка на плоскости и возвращает тот, длина которого больше. Использовать данную структуру. Написать функции ввода данных отрезка и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

9. Разработать структуру «Отрезок», отрезок задается координатами на плоскости его начала и конца. Написать функцию, которая вычисляет и возвращает длину отрезка. Написать функцию, которая определяет, пересекает ли отрезок оси координат. Использовать данную структуру. Написать функции ввода данных отрезка и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

10. Разработать структуру «Равносторонний треугольник», треугольник задается координатами его центра и длиной стороны. Написать функцию, которая вычисляет и возвращает периметр треугольника. Написать функцию, которая сравнивает два треугольника по площади и возвращает тот треугольник, площадь которого больше. Использовать данную структуру. Написать функции ввода данных треугольника и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

11. Разработать структуру «Треугольник», треугольник задается координатами его вершин. Написать функцию, которая вычисляет и возвращает периметр треугольника. Написать функцию, которая сравнивает периметры двух треугольников и возвращает тот треугольник, периметр которого больше. Использовать данную структуру. Написать функции ввода данных треугольника и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

12. Разработать структуру «Прямоугольник», прямоугольник задается координатами его верхнего левого и правого нижнего углов. Написать функцию, которая сравнивает периметры двух прямоугольников и возвращает тот прямоугольник, периметр которого больше. Написать функцию, которая вычисляет и возвращает сумму площадей двух прямоугольников. Использовать данную структуру. Написать функции ввода данных прямоугольника и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

13. Разработать структуру «Квадрат», квадрат задается координатами его центра и длиной стороны. Написать функцию, которая сравнивает периметры двух квадратов и возвращает тот квадрат, периметр которого больше. Написать функцию, которая вычисляет и возвращает сумму площадей трех квадратов. Использовать данную структуру. Написать функции ввода данных квадрата и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

14. Разработать структуру «Окружность», окружность задается координатами ее центра и радиусом. Написать функцию, которая сравнивает площади двух окружностей и возвращает ту окружность, площадь которой меньше. Написать функцию, которая вычисляет и возвращает сумму площадей трех окружностей. Использовать данную структуру. Написать функции ввода данных окружности и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

15. Разработать структуру «Ромб», ромб задается координатами его центра и длиной диагоналей. Написать функцию, которая вычисляет и возвращает площадь ромба. Написать функцию, которая сравнивает площади двух ромбов и возвращает тот ромб, площадь которого меньше. Использовать данную структуру. Написать функции ввода данных ромба и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

16. Разработать структуру «Окружность», окружность задается координатами ее центра и радиусом. Написать функцию, которая вычисляет и возвращает площадь круга. Написать функцию, которая определяет, захватывает ли окружность начало координат. Использовать данную структуру. Написать функции ввода данных окружности и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

17. Разработать структуру «Квадрат», квадрат задается координатами его центра и длиной стороны. Написать функцию, которая сравнивает два квадрата по площади и возвращает площадь того, который меньше. Написать функцию, которая определяет, центр какого квадрата находится ближе к началу координат, и возвращает его . Использовать данную структуру. Написать функции ввода данных квадрата и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

18. Разработать структуру «Окружность», окружность задается координатами ее центра и радиусом. Написать функцию, которая вычисляет и возвращает площадь круга. Написать функцию, которая определяет, пересекаются две окружности или нет. Использовать данную структуру. Написать функции ввода данных окружности и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

19. Разработать структуру «Окружность», окружность задается координатами ее центра и радиусом. Разработать структуру «Точка на плоскости». Написать функцию, которая вычисляет расстояние между двумя точками. Написать функцию, которая определяет, находится ли заданная точка внутри окружности . Использовать данные структуры. Написать функции ввода данных окружности и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.

20. Разработать структуру «Квадрат», квадрат задается координатами его центра и длиной стороны. Разработать структуру «Точка на плоскости». Написать функцию, которая вычисляет расстояние между двумя точками. Написать функцию, которая определяет, находится ли заданная точка внутри квадрата. Использовать данные структуры. Написать функции ввода данных квадрата и вывода на дисплей. Написать программу, демонстрирующую работу данных функций.