Управляющие инструкции. Указатели. Массивы.

лекция 6

План лекции

- Управляющие инструкции
  - Инструкции выбора if, switch
  - Инструкции цикла for, while, do-while
  - Инструкции перехода break, continue, goto и возврата return
- Указатели
  - Понятие указателя
  - Указатели в языке Си
  - Операции над указателями
  - Передача параметров функции по указателю
- Массивы
  - Массивы в языке Си
  - Связь массивов и указателей – генерация указателя
  - Описание массива в языке Си
  - Многомерные массивы
  - Массивы и строковые константы

Классификация инструкций языка Си

<инструкция> ::=
  <помеченная-инструкция>
| <инструкция-выражение>
| <составная-инструкция>
| <инструкция-выбора>
| <циклическая-инструкция>
| <инструкция-перехода>

Инструкции выбора if, switch

<инструкция-выбора> ::=
    'if' '(' <выражение> ')' <инструкция>
| 'if' '(' <выражение> ')' <инструкция> 'else' <инструкция>
| 'switch' '(' <выражение> ')' <инструкция>

Инструкции выбора -- switch

- Инструкция switch имеет следующий вид
  switch (выражение) {
  case константное-выражение : инструкции
  case константное-выражение : инструкции
  ...
  default: инструкции
  }
- Текст default: инструкции может отсутствовать
- Порядок работы
  - Вычисляется выражение в скобках, результат приводится к int
  - Если значение совпадает со значением одного из выражений после case, то управление передаётся на первую инструкцию после соотв. двоеточия. Дальнейшая работа зависит от этих инструкций
  - Иначе управление передается на первую инструкцию после default:

Операторы цикла (for, while, do-while)

<циклическая-инструкция> ::=
    'while' '(' <выражение> ')' <инструкция>
| 'do' <инструкция> 'while' '(' <выражение> ')'
| 'for' '(' [<выражение>] ';' [<выражение>] ';' [<выражение>] ')'
        <инструкция>

В цикле for любое из выражений может отсутствовать

Оператор цикла while

- Цикл while исполняет инструкцию до тех пор, пока выражение не станет равно 0

  while (выражение) инструкция

- выражение называется *условием продолжения цикла*
- инструкция называется *телом цикла*
- Значение выражение должно быть приводимым к типу int с помощью автоматических преобразований

Оператор цикла for

- Цикл for (в1; в2; в3) инструкция эквивалентен следующей последовательности инструкций с циклом while

```
в1;
while (в2) {
 инструкция
 в3;
}
```

Оператор цикла do-while

- Цикл do инструкция while (в2); эквивалентен следующим инструкциям

  инструкция
  while (в2)
  инструкция

Duff’s Device

```
send(to, from, count) //  Tom Duff in November 1983
register short *to, *from;
register count;
{
        register n = (count + 7) / 8;
        switch(count % 8) {
        case 0:       do {   *to++ = *from++; // вариант: *to++ = *from++;
        case 7:              *to++ = *from++;
        case 6:              *to++ = *from++;
        case 5:              *to++ = *from++;
        case 4:              *to++ = *from++;
        case 3:              *to++ = *from++;
        case 2:              *to++ = *from++;
        case 1:              *to++ = *from++;
                } while(--n > 0);
        }
}
```

Операторы перехода и возврата
break, continue, goto, return

<инструкция-перехода> ::=
    'goto' <идентификатор> ';'
| 'continue ';'
| 'break' ';'
| 'return' [<выражение>] ';'

Операторы перехода и возврата break, continue, return

- continue ;
  - Передаёт управление на проверку условия в while и do-while и на вычисление третьего выражения в for
  - Разрешено только в операторах цикла
- break ;
  - Передаёт управление на первый оператор после цикла или после оператора выбора
  - Разрешено в циклах и в операторе выбора switch
- return выражение ; и return ;
  - Завершает работу текущей функции и возвращает управление вызывающей функции
  - выражение должно быть приводимым к типу результата функции с помощью стандартных преобразований

Операторы перехода и возврата goto

- goto идентификатор ;
  - Передаёт управление на оператор, помеченный меткой идентификатор
  - Рекомендуется передавать управление только вперёд про тексту программы
  - Разрешено передавать управление из блока { } наружу за исключением выхода из функции
  - Нет смысла (но не запрещено) передавать управление внутрь блока { }
    - После такой передачи управления значения переменных, описанных внутри { }, неопределены
  - идентификатор должен быть меткой инструкции

- Управляющие инструкции
  - Инструкции выбора if, switch
  - Инструкции цикла for, while, do-while
  - Инструкции перехода goto, break, continue и возврата return
- Указатели
  - Понятие указателя
  - Указатели в языке Си
  - Операции над указателями
  - Передача параметров функции по указателю
- Массивы
  - Массивы в языке Си
  - Связь массивов и указателей – генерация указателя
  - Описание массива в языке Си
  - Многомерные массивы
  - Массивы и строковые константы

Понятие указателя

- Память ЭВМ делится на одинаковые ячейки -- байты
- Для обращения к ячейкам памяти процессор использует машинно-представимые целые числа без знака с максимальным числом битов – *адреса*
- Соответствие между адресами и ячейками памяти устанавливает ОС
  - Программа, работающая под управлением ОС, не может изменить это соответствие, но может изменять значения в ячейках памяти
  - Для программ память – линейный массив байтов
- Адреса, которым не соответствуют ячейки памяти, называются *недоступными* адресами или адресами недоступных ячеек памяти
- Адрес 0 является недоступным адресом по соглашению между программистами (в т.ч. авторами ОС) и разработчиками процессоров

Указатели в языке Си

- Указатель на (значения типа) T – это тип данных для работы с адресами значений типа T
- "Указатель на T" является *составным* типом от T
  - Составные типы получаются из простых типов char, int, и т.п. и других составных типов
- Тип "указатель на T" записывается в как "T*"

Указатели в языке Си -- примеры

- int *p;
  - Указатель на int
  - *p = 0 – ОК, p = 0 – ОК
- const int *p;
  - Указатель на const int
  - *p = 0 – ошибка
  - p = 0 – ОК
- int *const p;
  - Константа типа int*
  - *p = 0 – ОК
  - p = 0 -- ошибка
- int *p[];
  - Массив int*
  - *p[i] = 0, p[i] = 0 -- ОК
- const int *p[];
  - Массив указат. на const int
  - *p[i] = 0 – ошибка
  - p[i] = 0 -- ОК
- int *const p[];
  - Массив констант типа int*
  - *p[i] = 0 – ОК
  - p[i] = 0 -- ошибка
- const int *const p[];
  - Массив констант типа указатель на const int

Операции над указателями в Си

- NULL
  - Константа NULL -- адрес 0, отличный от всех других адресов
- &my_var
  - Результат – адрес первой из ячеек памяти, которые хранят значение переменной my_var
- *ptr_to_my_val
  - Результат – значение, на которое указывает ptr_to_my_val
  - *Разыменование* указателя
- ptr_to_my_struct->my_field
  - Результат – значение поля my_field структуры или объединения *ptr_to_my_struct

Операции над указателями в Си

- ptr1 == ptr2, ptr1 != ptr2
  - Проверка равенства адресов

- ptr1 < ptr2, ptr1 <= ptr2, ptr1 > ptr2, ptr1 >= ptr2
  - Проверка взаимного расположения в памяти ячеек с адресами ptr1 и ptr2
- ptr+N, N+ptr, ptr-N
  - Результат -- адрес ячейки, находящейся справа (+) или слева (-) на расстоянии N*sizeof(*ptr) байтов от ячейки по адресу ptr
  - Если ptr имеет тип void*, то ошибка компиляции

Операции над указателями в Си

- ptr1-ptr2
  - Результат -- расстояние между ячейками памяти по адресам ptr1 и ptr2, делённое на sizeof(*ptr1)
  - Если ptr1 и ptr2 имеют разны тип, то ошибка
  - Если ptr1 и ptr2 указывают не на элементы одного массива, то неопределённое поведение

- ptr[N], N[ptr]
  - Сокращение для *(ptr+N) и *(N+ptr)

Операции над указателями в Си

- ptr1 = ptr2, ptr1 += N, ptr2 -= N
  - Результат – ptr2
  - Побочный эффект – запись ptr2 в ptr1 до ближайшей точки следования
  - Результат доступа к памяти через ptr1 может неопределён, если ptr1 и ptr2 имеют разные типы

- ptr++, ptr--
  - Результат равен ptr
  - Побочный эффект ptr += 1 или ptr -= 1 до ближайшей точки следования
- ++ptr, --ptr
  - Результат равен ptr+1 или ptr-1
  - Побочный эффект ptr += 1 или ptr -= 1 до ближайшей точки следования

Передача параметров функции по указателю

- Пусть функция f вызывает функцию g и пусть var_in_f – переменная, описанная в f
- Поскольку тело g не пересекается с телом f, переменная var_in_f
  - Либо невидима в теле функции g
  - Либо скрыта одноимённой переменной, описанной в g
- Функция g не может ни прочитать, ни изменить значение переменной var_in_f в стековом кадре функции f используя идентификатор var_in_f, **НО**
- Функция g имеет возможность изменить значение var_in_f в стековом кадре f, если f передаст g в качестве параметра указатель &var_in_f на значение переменной var_in_f в стековом кадре f

Передача параметров функции по указателю -- пример

```
void my_swap_int(int x, int y)
{
    int old_x = x; // five, two невидимы
    x = y;
    y = old_x;
}
int main(void)
{
    int five = 5, two = 2;
    my_swap_int(five, two);
    // чему равно five? two?
    return 0;
}
```

Передача параметров функции по указателю -- пример

```
void my_swap_int(int *x, int *y)
{
    int old_x = *x;
    *x = *y;
    *y = old_x;
}
int main(void)
{
    int five = 5, two = 2;
    my_swap_int(&five, &two);
    return 0;
}
```

Передача параметров функции по указателю -- пример

```
void my_swap_int_ptr(int **px, int **py)
{
    int *old_px = *px;
    *px = *py;
    *py = old_px;
}
int main(void)
{
    int five=5, two=2, *pfive=&five, *ptwo=&two;
    my_swap_int_ptr(&pfive, &ptwo);
    // чему равно five? two? pfive? ptwo?
    return 0;
}
```

Указатели и передача аргументов функциям

```
void my_swap_int_ptr(const int **px, const int **py)
{
    // Почему не int *const *px и не int **const px??
    int *old_px = *px;
    *px = *py;
    *py = old_px;
}
int main(void)
{
    int five=5, two=2, *pfive=&five, *ptwo=&two;
    my_swap_int_ptr(&pfive, &ptwo);
    return 0;
}
```

- Массивы
  - Массивы в языке Си
  - Связь массивов и указателей – генерация указателя
  - Описание массива в языке Си
  - Многомерные массивы
  - Массивы и строковые константы

Массивы в языке Си

- Массив из (значений типа) T длины N – это тип данных для работы с набором из N значений типа T
  - N должно быть известно на момент компиляции (C89)
  - N должно быть известно на момент входа в блок, где описан массив (C99/C11)
- Массивы из T являются составными от типа T
  - Для разных N и одного T – разные типы
- Переменная A типа "массив из T длины N" описывается T A[N];
  - Без упоминания переменной -- T (*)[N]

Массивы в языке Си

- Значения элементов массива хранятся в памяти последовательно по возрастанию адресов
- Для A массива, описанного как T A[N], верно sizeof(A)==sizeof(T)*N==sizeof(A[0])*N
- Элементы массива длины N нумеруются от 0 до N-1

Связь массивов и указателей -- генерация указателя

- "Массивов в языке Си нет" (c)
- *Генерацией указателя* называется замена выражения A типа "массив из T" на *неизменяемый* указатель на A[0]
- Компилятор Си выполняет генерацию указателя всюду, где выражение типа массив *не является операндом* следующих операций
  - Унарные & и sizeof
    - ОК, ожидаемый результат
  - Унарные ++, --, левый операнд операций присваивания
    - Ошибка компиляции – почему?
  - Левый операнд операции . (точка)
    - Ошибка компиляции – почему?

Операции над массивами

- Генерация указателя позволяет выполнять над массивами те же операции, что и над указателями, кроме операций с побочным эффектом по отношению к операнду типа массив
  - Унарные ++, --
  - Операции присваивания, где массив -- левый операнд

Описание массива в языке Си

- T A[N];
  - Массив A из значений типа T длины N
- T A[N] = { И_0, И_1, ..., И_X };
  - Массив A из значений типа T длины N
  - X <= N-1
    - A[K] инициализируется с помощью И_K, K=0, …, X
    - Память, отведённая под A[X+1], …, A[N-1], заполняется байтом 0
  - X > N-1 – ошибка компиляции
- T A[] = { И_0, И_1, ..., И_X };
  - Массив A из значений типа T длины X+1
  - A[k] инициализируется с помощью И_k, k=0, …, X

Многомерные массивы

- Массив из T, где T – массив, называется многомерным массивом
- Примеры описания многомерных массивов
  - int A[10][100];
    - Массив из 10 массивов из 100 int
  - int A[2][2] = {{0, 1}, {2, 3}};
    - Массив из 2 массивов из 2 int
    - A[0][0] = 0, A[0][1] = 1, A[1][0] = 2, A[1][1] = 3

Многомерные массивы -- примеры

- int A[2][3];
  A[0] имеет тип int (*)[3]
  A[0][0] имеет тип int
  sizeof(A) = sizeof(A[0])*2
  sizeof(A[0]) = sizeof(A[0][0])*3

A[1][2] *(A[1]+2) *(*(A+1)+2)	A[1][1] *(A[1]+1) *(*(A+1)+1)	A[1][0] *(A[1]+0) *(*(A+1)+0) **(A+1)	A[0][2] *(A[0]+2) *(*(A+0)+2) *(*A+2)	A[0][1] *(A[0]+1) *(*(A+0)+1) *(*A+1)	A[0][0] *(A[0]+0) *(*(A+0)+0) **A

Направление роста адресов

Массивы и строковые константы

- Значением строковой константы длины N является инициализированный безымянный массив из N+1 char
- Для инициализации массива берутся последовательные символы из записи строковой константы
- После последнего символа из записи строковой константы берётся один символ '\0'
- Значения строковых констант хранятся в памяти глобальных пременных
- Значение строковой константы может начинаться или заканчиваться в середине значения другой строковой конст.

Массивы и строковые константы -- пример

- char my_str [] = "1234567890";
  // sizeof(my_str) == 11
  // эквивалентно
  // char my_str [] =
  // {'1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '\0'};

- Чему равно "1234"[0] ? "1234"[4] ?

- char *p = "1234";
  "1234"[0] = 'A';
  // значения строковых констант
  // могут занимать одни и те же ячейки памяти
  // p[0] равно либо 'A', либо '1'

Заключение

- Управляющие инструкции
  - Инструкции выбора if, switch
  - Инструкции цикла for, while, do-while
  - Инструкции перехода goto, break, continue и возврата return
- Указатели
  - Понятие указателя
  - Указатели в языке Си
  - Операции над указателями
  - Передача параметров функции по указателю
- Массивы
  - Массивы в языке Си
  - Связь массивов и указателей – генерация указателя
  - Описание массива в языке Си
  - Многомерные массивы
  - Массивы и строковые константы