Практическое занятие 12
Адресная арифметика

Пименов Евгений Сергеевич

Курс «Программирование»

Сибирский государственный университет телекоммуникаций и информатики (Новосибирск)

Осенний семестр, 2016

Целочисленные типы

Синтаксис объявления:

```
int a;

const int b = -42;

const int c = 0xFF;
```

Целочисленные типы

Представление в пямяти:

- Прямой код
- Дополнительный код

Знаковые целочисленные типы

Допустимые операции:

- Арифметические: `+ - * / %`
- Сравнения: `== != < <= > >=`
- Битовые: `| & ^ << >>`

Вещественные типы

Синтаксис объявления:

```
double a;

const double b = -42;

const double c = 10e5;
```

Вещественные типы

Допустимые операции:

- Арифметические: `+ - * /`
- Сравнения: `== != < <= > >=`

Вещественные типы

Представление в памяти:

IEEE 754

- Числа одинарной точности
- Числа двойной точности
- Числа тройной точности
- Числа четверной точности

Указатели

Синтаксис объявления:

```
      int *pfoo;
const int *pbar;
      int *const pbaz = &bar;
const int *const pquux = &quux;
```

Указатели

Допустимые операции:

- Разыменование: `*`
- Арифметические: `+ -`
- Сравнения: `== != < <= > >=`

Указатели

Представление в памяти: беззнаковое целое число

Разыменование


```
int a = 0;

int *p = &a;

*p = 42;
```

Разыменование


```
int a = 0;

int *p = &a;

*p = 42;
```

Переменная	p		a	
Адрес			0x10	0x14
Значение	0x10	...	42	

Арифметические операции


```
int *p = …; // адрес 0x10
++p;        // адрес?
```

Арифметические операции


```
int *p = …; // адрес 0x10
++p;        // адрес?
```

```
Адрес       | 0x10 | 0x14 | 0x18 | 0x1d |
            ^- p
```

Арифметические операции


```
int *p = …; // адрес 0x10
++p;        // адрес 0x14
```

```
Адрес       | 0x10 | 0x14 | 0x18 | 0x1d |
                    ^- p
```

Операции сравнения


```
int foo = 42, bar = 10, baz = 42;
int *pfoo = &foo, *pbar = &bar, *pbaz = &baz, *pquux = &foo;


pfoo == pbar;
pfoo == pbaz;
pfoo == pquux;
*pfoo == *pbar;
*pfoo == *pquux;
*pfoo == *pbaz;
```

Операции сравнения


```
int foo = 42, bar = 10, baz = 42;
int *pfoo = &foo, *pbar = &bar, *pbaz = &baz, *pquux = &foo;


pfoo == pbar;     // false
pfoo == pbaz;
pfoo == pquux;
*pfoo == *pbar;
*pfoo == *pquux;
*pfoo == *pbaz;
```

Операции сравнения


```
int foo = 42, bar = 10, baz = 42;
int *pfoo = &foo, *pbar = &bar, *pbaz = &baz, *pquux = &foo;


pfoo == pbar;     // false
pfoo == pbaz;     // false
pfoo == pquux;
*pfoo == *pbar;
*pfoo == *pquux;
*pfoo == *pbaz;
```

Операции сравнения


```
int foo = 42, bar = 10, baz = 42;
int *pfoo = &foo, *pbar = &bar, *pbaz = &baz, *pquux = &foo;


pfoo == pbar;     // false
pfoo == pbaz;     // false
pfoo == pquux;    // true
*pfoo == *pbar;
*pfoo == *pquux;
*pfoo == *pbaz;
```

Операции сравнения


```
int foo = 42, bar = 10, baz = 42;
int *pfoo = &foo, *pbar = &bar, *pbaz = &baz, *pquux = &foo;


pfoo == pbar;     // false
pfoo == pbaz;     // false
pfoo == pquux;    // true
*pfoo == *pbar;   // false
*pfoo == *pquux;
*pfoo == *pbaz;
```

Операции сравнения


```
int foo = 42, bar = 10, baz = 42;
int *pfoo = &foo, *pbar = &bar, *pbaz = &baz, *pquux = &foo;


pfoo == pbar;     // false
pfoo == pbaz;     // false
pfoo == pquux;    // true
*pfoo == *pbar;   // false
*pfoo == *pquux;  // true
*pfoo == *pbaz;
```

Операции сравнения


```
int foo = 42, bar = 10, baz = 42;
int *pfoo = &foo, *pbar = &bar, *pbaz = &baz, *pquux = &foo;


pfoo == pbar;     // false
pfoo == pbaz;     // false
pfoo == pquux;    // true
*pfoo == *pbar;   // false
*pfoo == *pquux;  // true
*pfoo == *pbaz;   // true
```

Передача массива в функцию


```
void print_int_array(int *begin, int *end);
```

```
      v- begin
    | 0 | 1 | 2 | 3 | 4 | - |
                          ^- end
```

Передача массива в функцию


```
void print_int_array(int *begin, int *end);
```

```
          v- begin
   | 0 | 1 | 2 | 3 | 4 | - |
                         ^- end
```

Передача массива в функцию


```
void print_int_array(int *begin, int *end);
```

```
                v- begin
   | 0 | 1 | 2 | 3 | 4 | - |
                          ^- end
```

Передача массива в функцию


```
void print_int_array(int *begin, int *end);
```

```
                  v- begin
   | 0 | 1 | 2 | 3 | 4 | - |
                          ^- end
```

Передача массива в функцию


```
void print_int_array(int *begin, int *end);
```

```
                    v- begin
  | 0 | 1 | 2 | 3 | 4 | - |
                        ^- end
```

Передача массива в функцию


```
void print_int_array(int *begin, int *end);
```

```
                          v- begin
  | 0 | 1 | 2 | 3 | 4 | - |
                          ^- end
```

Задание

Разработать функцию `string_suffix` .

Параметры:

- `[in] string` — строка
- `[in] suffix_length` — длина суффикса

Возвращаемое значение: указатель на суффикс указанной длины.

Задание

Пример:

```
$ ./string_suffix
Enter string: Hello.txt
Enter suffix length: 3

Suffix: txt
```

Домашнее задание

Разработать функцию `find_element` .

Параметры:

- `[in] begin` — указатель на нулевой элемент массива
- `[in] end` — указатель на элемент, расположенный за последним
- `[in] value` — значение искомого элемента

Возвращаемое значение: указатель на найденный элемент. Если элемент не найден, вернуть `NULL` .

Подготовить тестовое приложение. Выполнить поиск заданного элемента в массиве. Изменить его знак, используя возвращаемое значение функции `find_element`

Домашнее задание

Пример:

```
$ ./find_element
Enter array size: 5
Enter array elements: 5 4 3 4 5
Enter element to find: 4

Result: 5 -4 3 4 5

$ ./find_element
Enter array size: 5
Enter array elements: 5 4 3 4 5
Enter element to find: 42

Element 42 not found
Result: 5 4 3 4 5
```