Программирование

Лекция 3

Введение в массивы

Массив — это структура данных, которая содержит множество значений, относящихся к одному и тому же типу.

Для создания массива используется оператор объявления. Объявление массива должно описывать три аспекта:

- тип значений каждого элемента;
- имя массива;
- количество элементов в массиве.

Общая форма объявления массива:

```
имяТипа имяМассива[размерМассива];
```

```
short months[12];    // создает массив из 12 элементов типа short
```

Создание массива

```
int ragnar[7];
```

Массив, хранящий семь значений, каждое из которых является переменной типа **int**

Небольшие массивы целых чисел

```
// arrayone.cpp -- небольшие массивы целых чисел
#include <iostream>
int main()
{
    using namespace std;
    int yams[3];
    yams[0] = 7;
    yams[1] = 8;
    yams[2] = 6;

    int yamcosts[3] = {20, 30, 5};  // создание и инициализация массива
    // Примечание. Если ваш компилятор C++ не может инициализировать
    // этот массив, используйте static int yamcosts[3] вместо int yamcosts[3]

    cout << "Total yams = ";
    cout << yams[0] + yams[1] + yams[2] << endl;
    cout << "The package with " << yams[1] << " yams costs ";
    cout << yamcosts[1] << " cents per yam.\n";
    int total = yams[0] * yamcosts[0] + yams[1] * yamcosts[1];
    total = total + yams[2] * yamcosts[2];
    cout << "The total yam expense is " << total << " cents.\n";

    cout << "\nSize of yams array = " << sizeof yams;
    cout << " bytes.\n";
    cout << "Size of one element = " << sizeof yams[0];
    cout << " bytes.\n";
    return 0;
}
```

```
Total yams = 21
The package with 8 yams costs 30 cents per yam.
The total yam expense is 410 cents.

Size of yams array = 12 bytes.
Size of one element = 4 bytes.
```

Правила инициализации массивов

```
int cards[4] = {3, 6, 8, 10};      // все в порядке
int hand[4];                       // все в порядке
hand[4] = {5, 6, 7, 9};            // не допускается
hand = cards;                      // не допускается
```

```
float hotelTips[5] = {5.0, 2.5};
```

```
long totals[500] = {0};
```

```
short things[] = {1, 5, 3, 8};
```

Задача на массивы

- **Пример 1.** Задан массив A, содержащий 100 целых чисел. Найти сумму элементов этого массива.
- // сумма элементов массива A из 100 целых чисел
- int A[100] = {3,4,8,2,6,7,…,4,3};
- int summa; // переменная, содержащая сумму

Решение

- // Вычисление суммы
- summa = 0; // обнулить сумму
- for (int i=0; i<100; i++)
- summa += A[i];

- Перебор всех элементов массива выполняется в цикле for.
- Переменная summa сохраняет результирующее значение суммы элементов массива. Переменная i является счетчиком, определяющим индекс элемента массива A[i].

Задача на массивы

- **Пример 2.** Задан массив B, содержащий 20 вещественных чисел. Найти сумму элементов массива, которые лежат на парных позициях. Считать, что позиции 0, 2, 4 и т.д. есть парными.

- // сумма элементов массива B
- // лежащих на парных позициях
- float B[20] = {1,43,6,7,9,6,…,5,4,7};
- float sum; // переменная, содержащая сумму

Решение

- // Вычисление суммы
- sum = 0; // обнулить сумму
- for (i=0; i<20; i++)
- if ((i%2)==0)
- sum += B[i];

Строки

- Строка — это серия символов, сохраненная в расположенных последовательно байтах памяти.

```
char dog[8] = { 'b', 'e', 'a', 'u', 'x', ' ', 'I', 'I'};    // это не строка
char cat[8] = {'f', 'a', 't', 'e', 's', 's', 'a', '\0'};    // а это — строка

char bird[11] = "Mr. Cheeps";       // наличие символа \0 подразумевается
char fish[] = "Bubbles";            // позволяет компилятору подсчитать
                                    // количество элементов
```

Инициализация массива строкой

Конкатенация строковых литералов

```
cout << "I'd give my right arm to be" " a great violinist.\n";
cout << "I'd give my right arm to be a great violinist.\n";
cout << "I'd give my right ar"
"m to be a great violinist.\n";
```

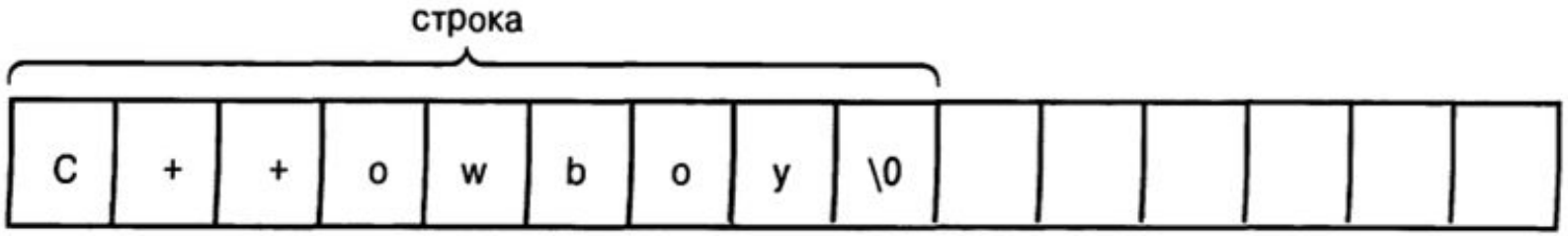
Использование строк в массивах

```
// strings.cpp -- сохране
#include <iostream>
#include <cstring>
int main()
{
    using namespace std;
    const int Size = 15;
    char name1[Size];                       // пустой массив
    char name2[Size] = "C++owboy";          // инициализация массива
    // ПРИМЕЧАНИЕ: некоторые реализации могут потребовать
    // ключевого слова static для инициализации массива name2

    cout << "Howdy! I'm " << name2;
    cout << "! What's your name?\n";
    cin >> name1;
    cout << "Well, " << name1 << ", your name has ";
    cout << strlen(name1) << " letters and is stored\n";
    cout << "in an array of " << sizeof(name1) << " bytes.\n";
    cout << "Your initial is " << name1[0] << ".\n";
    name2[3] = '\0';                        // установка нулевого символа
    cout << "Here are the first 3 characters of my name: ";
    cout << name2 << endl;
    return 0;
}
```

```
Howdy! I'm C++owboy! What's your name?
Basicman
Well, Basicman, your name has 8 letters and is stored
in an array of 15 bytes.
Your initial is B.
Here are the first 3 characters of my name: C++
```

Сокращение строки с помощью \0

```
const int Size = 15;
char name2[Size] = "C++owboy";
```




```
name2[3] = '\0';
```

Риски, связанные с вводом строк

```
#include <iostream>
int main()
{
    using namespace std;
    const int ArSize = 20;
    char name[ArSize];
    char dessert[ArSize];

    cout << "Enter your name:\n";                    // запрос имени
    cin >> name;
    cout << "Enter your favorite dessert:\n";        // запрос любимого десерта
    cin >> dessert;
    cout << "I have some delicious " << dessert;
    cout << " for you, " << name << ".\n";
    return 0;
}
```

```
Enter your name:
Alistair Dreeb
Enter your favorite dessert:
I have some delicious Dreeb for you, Alistair.
```

Строковый ввод с точки зрения с in

Построчное чтение ввода

- Класс istream: cin, getline () и get ().

Строчно-ориентированный ввод с помощью

```
cin.getline(name,20);
```

```
Enter your name:
Dirk Hammernose
Enter your favorite dessert:
Radish Torte
I have some delicious Radish Torte for you, Dirk Hammernose.
```

```
#include <iostream>
int main()
{
    using namespace std;
    const int ArSize = 20;
    char name[ArSize];
    char dessert[ArSize];

    cout << "Enter your name:\n";                    // запрос имени
    cin.getline(name, ArSize);                       // читать до символа новой строки
    cout << "Enter your favorite dessert:\n";        // запрос любимого десерта
    cin.getline(dessert, ArSize);
    cout << "I have some delicious " << dessert;
    cout << " for you, " << name << ".\n";
    return 0;
}
```

Строчно-ориентированный ввод с помощью get ()

```
cin.get(name, ArSize);
cin.get(dessert, Arsize);     // проблема
```

```
cin.get(name, ArSize);        // чтение первой строки
cin.get();                    // чтение символа новой строки
cin.get(dessert, Arsize);     // чтение второй строки
```

```
cin.get(name, ArSize).get();    // конкатенация функций
```

```
Enter your name:
Mai Parfait
Enter your favorite dessert:
Chocolate Mousse
I have some delicious Chocolate Mousse for you, Mai Parfait.
```

```
#include <iostream>
int main()
{
    using namespace std;
    const int ArSize = 20;
    char name[ArSize];
    char dessert[ArSize];

    cout << "Enter your name:\n";                 // запрос имени
    cin.get(name, ArSize).get();                  // читать строку и символ новой строки
    cout << "Enter your favorite dessert:\n";     // запрос любимого десерта
    cin.get(dessert, ArSize).get();
    cout << "I have some delicious " << dessert;
    cout << " for you, " << name << ".\n";
    return 0;
}
```

Введение в класс string

```
#include <iostream>
#include <string> // обеспечение доступа к классу string
int main()
{
    using namespace std;
    char charr1[20];
    char charr2[20] = "jaguar";
    string str1;
    string str2 = "panther";

    cout << "Enter a kind of feline: ";
          // Введите животное из семейства кошачьих
    cin >> charr1;
    cout << "Enter another kind of feline: ";
          // Введите другое животное из семейства кошачьих
    cin >> str1;                              // использование cin для ввода
    cout << "Here are some felines:\n";
    cout << charr1 << " " << charr2 << " "
         << str1 << " " << str2               // использование cout для вывода
         << endl;
    cout << "The third letter in " << charr2 << " is "
         << charr2[2] << endl;
    cout << "The third letter in " << str2 << " is "
         << str2[2] << endl;                  // использование нотации массивов
    return 0;
}
```

```
Enter a kind of feline: ocelot
Enter another kind of feline: tiger
Here are some felines:
ocelot jaguar tiger panther
The third letter in jaguar is g
The third letter in panther is n
```

Присваивание, конкатенация и добавление

```
char charr1[20];           // создание пустого массива
char charr2[20] = "jaguar"; // создание инициализированного массива
string str1;               // создание пустого объекта string
string str2 = "panther";   // создание инициализированной строки
charr1 = charr2;           // НЕ ПРАВИЛЬНО, присваивание массивов не разрешено
str1 = str2;               // ПРАВИЛЬНО, присваивание объектов допускается
```

```
string str3;
str3 = str1 + str2;        // присвоить str3 объединение строк
str1 += str2;              // добавить str2 в конец str1
```

Дополнительные сведения об операциях класса string

```
strcpy(charr1, charr2);  // копировать charr2 в charr1
strcat(charr1, charr2);  // добавить содержимое charr2 к char1
```

Дополнительные сведения об операциях класса string

```
#include <iostream>
#include <string>              // обеспечение доступа к классу string
#include <cstring>             // библиотека обработки строк в стиле C
int main()
{
    using namespace std;
    char charr1[20];
    char charr2[20] = "jaguar";
    string str1;
    string str2 = "panther";

    // Присваивание объектов string и символьных массивов
    str1 = str2;                          // копирование str2 в str2
    strcpy(charr1, charr2);               // копирование charr2 в charr1

    // Добавление объектов string и символьных массивов
    str1 += " paste";                     // добавление " paste" в конец str1
    strcat(charr1, " juice");             // добавление " juice" в конец charr1

    // Определение длины объекта string и строки в стиле C
    int len1 = str1.size();               // получение длины str1
    int len2 = strlen(charr1);            // получение длины charr1
    cout << "The string " << str1 << " contains "
         << len1 << " characters.\n";
    cout << "The string " << charr1 << " contains "
         << len2 << " characters.\n";
    return 0;
}
```

```
The string panther paste contains 13 characters.
The string jaguar juice contains 12 characters.
```

Дополнительные сведения об операциях класса string

```
str3 = str1 + str2;
```

```
strcpy(charr3, charr1);
strcat(charr3, charr2);
```

```
char site[10] = "house";
strcat(site, " of pancakes");      // проблема с нехваткой памяти
```

```
int len1 = str1.size();            // получение длины str1
int len2 = strlen(charr1);         // получение длины charr1
```

Задача на поиск символа в строке

- **Пример 3.** Задана строка символов. Определить, есть ли заданный символ с в этой строке символов.

Решение

```
char S[50]; // строка символов
char c; // искомый символ
int i;
bool f_is; // f_is=true - символ есть в строке, иначе f_is=false
// ввод строки S
// ...
// ввод символа c
// ...
for (i=0; i<strlen(S); i++)
    if (S[i]==c)
    {
        f_is = true; // символ c есть в строке S
        break;
    }
if (f_is)
    cout<< "Символ " + c.ToString() + " есть в строке";
else
    cout<< "Символа " + c.ToString() + "нет в строке";
```

Задача на строки

- **Пример 4.** Пусть задан некоторый текст. Вычислить, сколько раз повторяется наперед заданный символ **a**
- // нахождение числа вхождений символа в строке
- char S[50] = “example”; // строка символов
- char a = ‘a’; // заданный символ

Решение

- int i;
- int k; // результат - число вхождений символа a в строке S
- k = 0; // в начале обнулить счетчик k
- for (i=0; i<strlen(S); i++)
- if (S[i]==a)
- k++; // увеличить счетчик на 1

Задача на замену символов

- **Пример 1.** В заданном тексте заменить все символы ‘+’ на ‘ — ‘.
- // замена символов
- char str[50]; // заданный текст
- int i;
- for (i=0; i<strlen(str); i++)
- if (str[i] == '+')
- str[i] = '-';

Введение в структуры

- Структура — более универсальная форма данных, нежели массив, потому что одна структура может хранить элементы более чем одного типа.
- Структура представляет собой определяемый пользователем тип с объявлением, описывающим свойства данных типа.
- Создание структуры — процесс, состоящий из двух частей. Вначале определяется описание структуры, в котором перечисляются и именуются типы данных, хранящиеся в структуре. Затем создаются структурные переменные.

```
struct inflatable   // объявление структуры
{
    char  name[20];
    float volume;
    ouble price;
};
```

Части описания структуры


```
inflatable hat;              // hat – структурная переменная типа inflatable
inflatable woopie_cushion;   // переменная типа inflatable
inflatable mainframe;        // переменная типа inflatable
```

Использование структур в программах

```
#include <iostream>
struct inflatable                    // объявление структуры
{
    char name[20];
    float volume;
    double price;
};
int main()
{
    using namespace std;
    inflatable guest =
    {
        "Glorious Gloria",           // значение name
        1.88,                        // значение volume
        29.99                        // значение value
    }; // guest – структурная переменная типа inflatable
    // Инициализация указанными значениями
    inflatable pal =
    {
        "Audacious Arthur",
        3.12,
        32.99
    }; // pal – вторая переменная типа inflatable
    cout << "Expand your guest list with " << guest.name;
    cout << " and " << pal.name << "!\n";    // pal.name - член name переменной pal
    cout << "You can have both for $";
    cout << guest.price + pal.price << "!\n";
    return 0;
}
```

```
Expand your guest list with Glorious Gloria and Audacious Arthur!
You can have both for $62.98!
```

Прочие свойства структур

```
#include <iostream>
struct inflatable
{
    char name[20];
    float volume;
    double price;
};
int main()
{
    using namespace std;
    inflatable bouquet =
    {
        "sunflowers",
        0.20,
        12.49
    };
    inflatable choice;
    cout << "bouquet: " << bouquet.name << " for $";
    cout << bouquet.price << endl;
    choice = bouquet;           // присваивание одной структуры другой
    cout << "choice: " << choice.name << " for $";
    cout << choice.price << endl;
    return 0;
}
```

```
bouquet: sunflowers for $12.49
choice: sunflowers for $12.49
```

Прочие свойства структур

```
struct perks
{
    int key_number;
    char car[12];
} mr_smith, ms_jones;     // две переменных типа perks
```

```
struct perks
{
    int key_number;
    char car[12];
} mr_glitz =
{
    7,                    // значение члена mr_glitz.key_number
    "Packard"             // значение члена mr_glitz.car
};
```

```
struct                    // дескриптора нет
{
    int x;                // два члена
    int y;
} position;               // структурная переменная
```

Массивы структур

```
inflatable gifts[100];          // массив из 100 структур inflatable
```

```
cin >> gifts[0].volume;         // используется член volume первой структуры
cout << gifts[99].price << endl; // отображается член price последней структуры
```

```
inflatable guests[2] =          // инициализация массива структур
{
    {"Bambi", 0.5, 21.99},      // первая структура в массиве
    {"Godzilla", 2000, 565.99}  // следующая структура в массиве
};
```

Массивы структур

```
#include <iostream>
struct inflatable
{
    char   name[20];
    float  volume;
    double price;
};
int main()
{
    using namespace std;
    inflatable guests[2] =          // инициализация массива структур
    {
        {"Bambi", 0.5, 21.99},      // первая структура в массиве
        {"Godzilla", 2000, 565.99}  // следующая структура в массиве
    };
    cout << "The guests " << guests[0].name << " and " << guests[1].name
         << "\nhave a combined volume of "
         << guests[0].volume + guests[1].volume << " cubic feet.\n";
    return 0;
}
```

```
The guests Bambi and Godzilla
have a combined volume of 2000.5 cubic feet.
```

Задача на массивы структур

```
#include <iostream>
using namespace std;

struct PlayerInfo {
    int skill_level;
    string name;
};
using namespace std;

int main() {
    // как и с обычными типами, вы можете объявить массив структур
    PlayerInfo players[5];
    for (int i = 0; i < 5; i++) {
        cout << "Please enter the name for player : " << i << '\n';
        // сперва получим доступ к элементу массива, используя
        // обычный синтаксис для массивов, затем обратимся к полю структуры
        // с помощью точки
        cin >> players[ i ].name;
        cout << "Please enter the skill level for " << players[ i ].name << '\n';
        cin >> players[ i ].skill_level;
    }
    for (int i = 0; i < 5; ++i) {
        cout << players[ i ].name << " is at skill level " << players[i].skill_level << '\n';
    }
}
```

Объединения

- Объединение — это формат данных, который может хранить в пределах одной области памяти разные типы данных, но в каждый момент времени только один из них.

```
union one4all
{
    int int_val;
    long long_val;
    double double_val;
};


one4all pail;
pail.int_val = 15;                  // сохранение int
cout << pail.int_val;
pail.double_val = 1.38;             // сохранение double, int теряется
cout << pail.double_val;
```

Объединения

```
struct widget
{
    char brand[20];
    int type;
    union id                        // формат зависит от типа предмета
    {
        long id_num;                // предметы первого типа
        char id_char[20];           // прочие предметы
    } id_val;
};
...
widget prize;
...
if (prize.type == 1)                // оператор if-else (глава 6)
    cin >> prize.id_val.id_num; // использование поля name для указания режима
else
    cin >> prize.id_val.id_char;
```

Анонимное объединение

- Анонимное объединение не имеет имени; в сущности, его члены становятся переменными, расположенными по одному и тому же адресу в

```
struct widget
{
    char brand[20];
    int type;
    union                          // анонимное объединение
    {
        long id_num;               // предметы первого типа
        char id_char[20];          // прочие предметы
    };
};
...
widget prize;
...
if (prize.type == 1)
    cin >> prize.id_num;
else
    cin >> prize.id_char;
```

Перечисления

```
enum spectrum {red, orange, yellow, green, blue, violet, indigo, ultraviolet};

spectrum band;              // band – переменная типа spectrum

band = blue;                // правильно, blue - перечислитель
band = 2000;                // неправильно, 2000 – не перечислитель

band = orange;              // правильно
++band;                     // неправильно
band = orange + red;        // неправильно
...
```

Перечисления — целочисленные типы, и они могут быть представлены в виде
int, однако тип int не преобразуется автоматически в тип перечисления:

```
int color = blue;           // правильно, тип spectrum приводится к int
band = 3;                   // неправильно, int не преобразуется в spectrum
color = 3 + red;            // правильно, red преобразуется в int
...
```

Указатели

- **Указатели** представляют собой переменные, хранящие адреса значений вместо самих значений.
- Например, если home — переменная, то &home

```
// address.cpp -- использование операции & для нахождения адреса
#include <iostream>
int main()
{
    using namespace std;
    int donuts = 6;
    double cups = 4.5;

    cout << "donuts value = " << donuts;
    cout << " and donuts address = " << &donuts << endl;
    // ПРИМЕЧАНИЕ: может понадобиться использовать
    // unsigned (&donuts) и unsigned (&cups)
    cout << "cups value = " << cups;
    cout << " and cups address = " << &cups << endl;
    return 0;
}
```

```
donuts value = 6 and donuts address = 0x0065fd40
cups value = 4.5 and cups address = 0x0065fd44
```

Указатели и философия C++

- Объектно-ориентированное программирование (ООП) отличается от традиционного процедурного программирования в том, что ООП делает особый акцент на принятии решений во время выполнения вместо времени компиляции. Время выполнения означает период работы программы, а время компиляции — период сборки программы компилятором в единое целое.

- Применяя операцию *, называемую **косвенным значением** или **операцией разыменования**, можно получить значение, хранящееся в указанном месте.

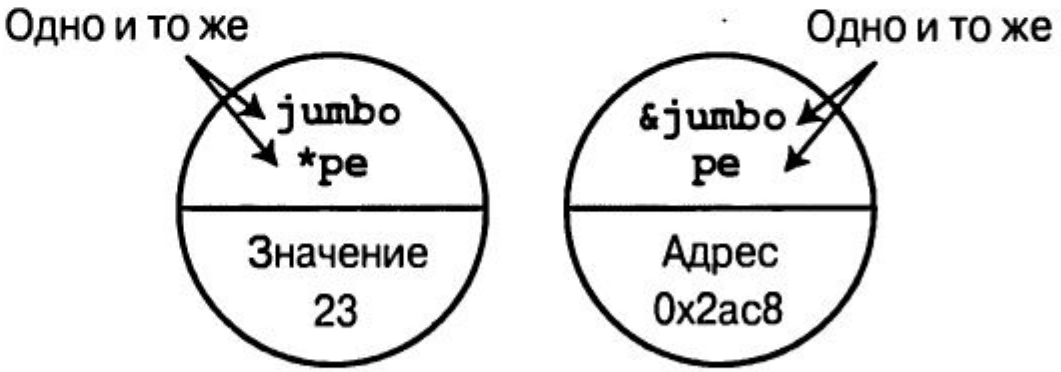
Указатели

```
#include <iostream>
int main()
{
    using namespace std;
    int updates = 6;                    // объявление переменной
    int * p_updates;                    // объявление указателя на int

    p_updates = &updates;               // присвоить адрес int указателю

    // Выразить значения двумя способами
    cout << "Values: updates = " << updates;
    cout << ", *p_updates = " << *p_updates << endl;

    // Выразить адреса двумя способами
    cout << "Addresses: &updates = " << &updates;
    cout << ", p_updates = " << p_updates << endl;

    // Изменить значение через указатель
    *p_updates = *p_updates + 1;
    cout << "Now updates = " << updates << endl;
    return 0;
}
```

```
Values: updates = 6, *p_updates = 6
Addresses: &updates = 0x0065fd48, p_updates = 0x0065fd48
Now updates = 7
```

Две стороны одной монеты

```
int jumbo = 23;
int * pe = &jumbo;
```



Выделение памяти с помощью операции new

*имяТипа * имя_указателя* = new *имяТипа*;

```
#include <iostream>
int main()
{
    using namespace std;
    int nights = 1001;
    int * pt = new int;
    *pt = 1001;
    cout << "nights value = ";                          // значение nights
    cout << nights << ": location " << &nights << endl; // расположение nights
    cout << "int ";                                     // значение и расположение int
    cout << "value = " << *pt << ": location = " << pt << endl;
    double * pd = new double;   // выделение пространства для double
    *pd = 10000001.0;           // сохранение в нем значения double
    cout << "double ";
    cout << "value = " << *pd << ": location = " << pd << endl;
          // значение и расположение double
    cout << "location of pointer pd: " << &pd << endl;
          // расположение указателя pd
    cout << "size of pt = " << sizeof(pt);                  // размер pt
    cout << ": size of *pt = " << sizeof(*pt) << endl;     // размер *pt
    cout << "size of pd = " << sizeof pd;                   // размер pd
    cout << ": size of *pd = " << sizeof(*pd) << endl;     // размер *pd
    return 0;
}
```

```
nights value = 1001: location 0028F7F8
int value = 1001: location = 00033A98
double value = 1e+007: location = 000339B8
location of pointer pd: 0028F7FC
size of pt = 4: size of *pt = 4
size of pd = 4: size of *pd = 8
```

Освобождение памяти с помощью операции delete

```
int * ps = new int;   // выделить память с помощью операции new
...                   // использовать память
delete ps;            // по завершении освободить память
                      // с помощью операции delete
```

```
int * ps = new int;   // нормально
delete ps;            // нормально
delete ps;            // теперь не нормально!
int jugs = 5;         // нормально
int * pi = &jugs;     // нормально
delete pi;            // не допускается, память не была выделена new
```

Вопросы

- 1. Как вы объявите следующие объекты данных?
- а. actor — массив из 30 элементов char.
- б. betsie — массив из 100 элементов short.
- в. chuck — массив из 13 элементов float.
- г. dipsea — массив из 64 элементов long double.
- 2. Объявите массив из пяти элементов int и инициализируйте его первыми пятью положительными нечетными числами.
- 3. Напишите оператор, который присваивает переменной even сумму первого и последнего элементов массива из вопроса 2.
- 4. Разработайте объявление структуры, описывающей рыбу. Структура должна включать вид, вес в полных унциях и длину в дробных дюймах.