Занятие 3

Составные типы

Массивы. Строки в стиле C. Строки класса string. Чтение строк. Структуры. Инициализация структур.

Пример 1. *Небольшие массивы целых чисел.*
Массивы используются для хранения нескольких переменных одного типа. Массивы целых чисел декларируются как

```
int iarr[3]; //массив из трех элементов
```

при этом выделяется память для хранение 3-х переменных, но не происходит инициализация, а значит в этом участке памяти может быть записана какая-либо информация. Чтение этой информации приводит к ошибке. Элементы массива можно сразу инициализировать, например,

```
int iarr[3]={1,2,3};
int iarr2[3]={0}; // все нули
```

В качестве размера массива должна использоваться постоянная, декларируемая через команду препроцессора `#define` или с `const`, например,

```
const int N=3;
int irr[N];
```

Некоторые компиляторы поддерживают создание массивов, где размер не является постоянной, например,

```
int N=3;
int irr[N];
```

Так как использование массивов с переменной длиной (variable length arrays) не поддерживается стандартом и может вызывать ошибки при компиляции, то такие декларации следует избегать. Для работы с массивами переменной длины следует использовать стандартные функции языка C (`malloc, calloc`) или C++ (`new`).

Для доступа к отдельному элементы массива тоже используются квадратные скобки. При этом индексация начинается всегда с 0 и заканчивается $N-1$, где N-количество элементов. Таким образом, если массив декларирован как

```
int iarr[3]={1,2,3};
```

то команда

```
cout << iarr[3];
```

приведет к ошибке, так как будет осуществляться чтение из участка памяти за пределами массива. Компиляторы не проверяют наличие таких ошибок. Пример ниже иллюстрирует работу с массивами.

```cpp
#include <iostream>
int main(void){
  using namespace std;
  int iarr[3]; // Создание (объявление) массива из трех элементов
  iarr[0] = 7; // Присваивание значения первому элементу
  iarr[1] = 8;
  iarr[2] = 6;

  int iarr2[3] = {20, 30, 5}; // Создание и инициализация массива
  cout << iarr[0]  + iarr[1]  + iarr[2]  << endl; // Напечатать сумму
  cout << iarr2[0] + iarr2[1] + iarr2[2] << endl;
  return 0;
}
```

Пример 2. *Строки в стиле C. Инициализация строк. Определение длины строки.* Строка в C – это последовательность символов, которая заканчивается нулевым символом (0 или '\0'). Последовательность символов хранится в массиве типа `char`. В примере ниже показаны примеры задания строк и работы с ними.

```cpp
#include <iostream>
#include <cstring> // для функции strlen()
int main(void) {

  using namespace std;
  const int ArSize = 20;

  char name1[ArSize] = "Hello, world!"; // инициализация постоянной строкой
  cout << "name1 = " << name1;          // вывод строки на экран
  cout << endl;
  cout << "sizeof(name1)=" << sizeof(name1); // размер массива
  cout << endl;
  cout << "strlen(name1)=" << strlen(name1);  // длина строки
  cout << endl;

  // char name2[10]={'a','b','c'};      // остальные элементы будут 0
  // char name2[10]={'a','b','c','\0'}; // записали 0 в явном виде
  char name2[10]={'a','b','c', 0};      // записали 0 в явном виде
  cout << "name2=" << name2;
  cout << endl;
  cout << "strlen(name2)=" << strlen(name2);  // длина строки
  cout << endl;

  // char name3[5]="Hello"; // ошибка при компиляции

  char name4[5]={'h', 'e', 'l', 'l', 'o'};  // нет ошибки, но это не строка!
  cout << "name4=" << name4; // может вывести мусор
  cout << endl;
  cout << "strlen(name4)=" << strlen(name4);// может вывести не 5
  cout << endl;

  //  char name5[10]={104,101,108,108,111};     // строка
  char name5[10]={104,101,108,108,111,0};      // строка

```

```
  cout << "name5=" << name5;
  cout << endl;
  cout << "strlen(name5)=" << strlen(name5); // выведет 5
  cout << endl;

  // можно получить доступ к отдельному символу
  cout << "name5[0]=" << name5[0]; // распечатали первый символ
  cout << endl;

  // можно поменять символы
  name5[0]='H';  // изменили один символ
  name5[5]='!';  // изменили еще один символ
  name5[6]='\0'; // и записали 0 в конце

  cout << "name5=" << name5;
  cout << endl;

  return 0;
}
```

Пример 3. *Использование класса C++ string.*
Для удобства работы с текстом в C++ чаще всего используются объекты класса `string` вместо массивов символов. Объекты `string` не нуждаются в задании размеров, так как нужный размер памяти выделяется автоматически (это свойство класса). Для доступа к отдельным символам объекта типа `string` также используются квадратные скобки, внутри которых указывается индекс требуемого символа.

```
#include <iostream>
#include <string>   // обеспечение доступа к классу string

int main(void) {
  using namespace std;

  string email_suffix = "rf.unn.ru";// объявление и инициализация объекта
  string dean_name = "matrosov";    // строкой в C стиле
  string student_name;              // объявление пустого объекта string

  cout << "Enter your last name to create email address: ";
  cin  >> student_name;

  cout << "The Dean of the Radiophysics faculty is " << dean_name << endl;

  // использование [] для получение доступа к символу в строке
  cout << "\nThe second letter in \"" << student_name
       << "\" is \'" << student_name[1] << "\'" << endl;

  cout << "\nNow let's make emails for the Dean and for you!\n";

  string    dean_email = dean_name    + "@" + email_suffix; // Конкатенация
  string student_email = student_name + "@" + email_suffix; // (сложение)

  cout << "Dean\'s email is " << dean_email << endl;
  cout << "Your email is " << student_email << endl;

  // получение длины строки с помощью метода size ()
  cout << "\nYour email address is " << student_email.size()
```

```
30        << " symbols long\n";
31
32    return 0;
33 }
```

Пример 4. *Чтение более одного слова с помощью методов* `getline()`.
Последовательность операций

```
string name;
cin >> name;
```

запишет в строку `name` символы, введенные пользователем, до первого пробельного символа. Пробельные символы включают `' '` (пробел), `'\t'` (горизонтальная табуляция), `'\n'` (новая строка), `'\v'` (вертикальная табуляция). Все остальное останется в потоке ввода и будет влиять на последующий ввод.

Программа ниже будет работать только, если вводимое имя состоит из одного слова.

```
1  #include <iostream>
2  #include <cstring>
3  int main(void) {
4    using namespace std;
5
6    string name1;
7
8    cout << "Enter your name:\n";
9    cin >> name1;
10   cout << "Hello, " << name1 << "!" << endl;
11
12   return 0;
13 }
```

Для ввода строки, содержащей пробельные символы, можно использовать команду `getline()`. В примере ниже `getline()` читает символы до символа новой строки `'\n'`. Также можно задавать символ, до которого происходит чтение.

```
1  #include <iostream>
2  #include <cstring>
3  int main(void) {
4    using namespace std;
5
6    string name1;
7
8    cout << "Enter your full name:\n";
9    getline(cin, name1); // чтение в строку
10   cout << "Hello, " << name1 << "!" << endl;
11
12   return 0;
13 }
```

Заметим, что команда `getline()` может быть использована и со строками типа `char`. Функция `cin.getline(char *name, int N)` записывает символы из входного потока в

массив `name` пока не встретится символ новой строки `'\n'` или не считается `N-1` символов. Добавляет символ окончания строки `'\0'` в массив и убирает символ новой строки из потока ввода.

Комбинирование команд `cin` и `getline()` может приводить к неожиданным результатам. Команда `cin` игнорирует пробельные символы и оставляет в потоке ввода символ `'\n'`. Этот символ будет влиять на работу последующего вызова `getline()`. Поэтому требуется все начальные пробельные символы убрать, см. пример ниже.

```
#include <iostream>
#include <cstring>
int main(void) {
  using namespace std;

  int favorite_number;
  cout << "Enter your favorite number\n";
  cin  >> favorite_number;

  string name;
  cout << "Enter your full name:\n";
  //getline(cin, name); // работать не будет
  getline(cin >> ws, name); // убираем пробельные символы

  cout << name << ", your favorite number is " << favorite_number << "." <<
   endl;

  return 0;
}
```

Пример 5. *Создание простых структур.*
Структуры используются для совместного хранения нескольких переменных разного типа. Пример ниже иллюстрирует создание структур и работу с ними.

```
#include <iostream>
#include <string>

struct movie {// определение структуры
    std::string name;  // используем string из пространства имен std
    int year;
    int duration;
};

int main(void) {
    using namespace std;

    movie m1 = {"Interstellar", 2014, 169};// объявление и инициализация,
    movie m2 = {.name = "The Godfather",    // порядок элементов как в
                .year = 1972,               // определении.
                .duration = 175};           // Использование имен переменных
                                            // не является стандартом в C++.

    movie m3; // объявление без инициализации
    cout << "Enter your favourite movie" << endl << "Name: ";
    getline(cin, m3.name);  // читает все символы до конца строки
    cout << "Year: ";
    cin >> m3.year;
    cout << "Duration: ";
```

```
25      cin >> m3.duration;
26
27      cout << "\nList of movies on your shelf: \n"
28           << "1) " << m3.name
29           << " ("  << m3.duration << " min.)\n"
30           << "2) " << m2.name
31           << " ("  << m2.duration << " min.)\n"
32           << "3) " << m1.name
33           << " ("  << m1.duration << " min.)\n";
34      return 0;
35  }
```

Задания

Задание 3.1. Напишите программу, которая приглашает пользователя ввести почтовый адрес, состоящий из индекса, названия города, названия улицы, номера дома и квартиры, и ФИО получателя (в отдельных переменных). А затем программа должна из имеющихся данных составить 3 строки соответствующие следующему форматированному выводу:

```
603950, Nizhny Novgorod
Gagarina ave., 3 - 33
Ivanov Ivan Ivanovich
```

и вывести на экран. Используйте объекты `string` и методы из заголовочного файла `string`. Для чтения более одного слова можно воспользоваться функцией `getline(cin, name_of_string)`.

Задание 3.2. Напишите программу, в которой задан числовой массив из 10 элементов. Программа должна вывести этот массив на экран, затем запросить у пользователя ввод произвольного номера (индекс) элемента и вывести на экран квадратный корень из элемента с введенным индексом.

Задание 3.3. Предположим, Вам надо вести учет книг в книжном магазине. Для каждой книги Вам нужна следующая информация:

- автор,
- название,
- количество страниц,
- цена.

Придумайте структуру для хранения данных. Напишите программу, где создаётся одна переменная этого типа и заполняется данными, запрашивая их у пользователя последовательно. Полученная информация затем выводится на экран. Продемонстрируйте работу программы, используя книгу: Margaret Mitchell, “Gone with the Wind”, 833 стр., $25.99.