ЕН.Ф.02 – Информатика и программирование

Лекция 1. Введение в язык программирования C++

Конова Елена Александровна
E_Konova@mail.ru

Некоторые определения

Алгоритм (программа) – последовательность инструкций компьютеру, которые управляют его работой по обработке данных.

Две важные составляющие:

назначение – обработка данных любых форматов представления;

механизм – программное управление (принцип Фон Неймана)

При работе программы программа (или ее часть) и обрабатываемые ею данные (или их часть) находятся в ОЗУ.

Требования к алгоритму

1. Детерминированность – однозначное толкование любого шага алгоритма.
2. Результативность – отсутствие тупиковых ситуаций на любых наборах входных данных.

Из любой нестандартной ситуации должен быть выход, например, деление на 0 должно вызывать обработчик ошибок. Инструменты – исключения.

3. Массовость – алгоритм предназначен для решения некоторого класса задач.
4. Требования к программе: удобный (дружественный) интерфейс (человеческое лицо).

Некоторые определения

Язык программирования – инструмент для создания программ.

Формальные языки: C++, Basic, Pascal, Java, ets...

Различаются по парадигме программирования:

- процедурно ориентированные;
- объектно-ориентированные;
- логико-ориентированные;
- и другие.

Основные характеристики, исходя из назначения:

- ограниченный набор изобразительных средств;
- ограниченное количество правил.

Состав изобразительных средств

Исходя из назначения языка.

Алфавит → Лексемы → Операторы → Программы → Проекты.

В алфавит включены все используемые знаки.

Лексемы – слова и выражения.

Операторы – предложения для описания одного шага алгоритма.

Программы – совокупность блоков, функций, файлов.

Проекты – совокупность программ для решения прикладной задачи.

Состав изобразительных средств

Все правила сводятся к двум группам.

Синтаксис – формальный набор правил, определяющий способ построения любых конструкций языка.

Например,

y=pow(x,2)/(c+d);

Семантика – множество правил, определяющих смысл синтаксических конструкций.

Механизм – технология выполнения, заложенная в компиляторе.

Например, pow, это возведение в степень, выражение вычисляется слева направо, знак = , это присваивание.

Алфавит

Алфавит – набор разрешенных символов языка.

Четыре группы символов.

1. Буквы

Используются латинские буквы `a..z` `A..Z`

```
int a;// Это разные имена.
int A;
```

Русские буквы не входят в алфавит и используются ограниченно:

а) в комментариях;

б) в текстовых константах.

Например, `"Jonn Braun"`

Алфавит

2. Цифры

Используются цифры 0-9.

3. Специальные символы

а) парные символы: "" " { } [ ] ( ) ets…

б) знаки операций: + – * / % ^ < ! ets…

в) знаки препинания: . , ; ets…

г) прочие: : ? < = > _ & * # ~ ^ .

4. Невидимые символы

Пробел, разделитель строк, табулятор и прочие.

Особенность – код есть, а не видны в редакторе.

Замечание: деление условно, классификация может быть различной.

Лексемы

Лексема – единица текста (конструкция, слово), воспринимаемая компилятором как единое неделимое целое

Различают 5 классов лексем

1. Имена (идентификаторы) для именования объектов программы.

```
My_File,   x1,  alpha
```

2. Служебные (ключевые) слова, конструкции языка.

```
for  while  do  if enum  int  double
```

3. Константы

```
-1    12.5E-2    "Miranda"    '!'
```

4. Операции (знаки операций) == >= != >>

5. Разделители (знаки пунктуации) " " " [ ] { } ( )

Основные понятия языка

Далее в плане

- Данные.
- Концепция типов.
- Представление данных.
- Константы и переменные.

Данные

Данные – все, что подлежит обработке с помощью программы независимо от формата представления (числовые данные, текст, графика, звук и прочие).

Классификация данных:

а) по типам – базовые и конструируемые.

Базовые типы, это встроенные в реализацию языка, предопределенные реализацией.

б) по способу организации – независимо от типа, данные могут быть постоянными (константы) и переменными (переменные).

Константы

Константа – данное, которое не меняет значения при выполнении программы и присутствует в ней явно.

Тип значения константы определен записью, например:

`-9.9` – числовое значение с плавающей точкой.

`12` – целочисленное десятичное значение.

`'1'` – символьная константа.

`"Text"` – строковая константа.

Именованные константы имеют синтаксис:

`const` Тип Имя = Значение;

```
const int a=5;  // Тип int,значение =5,имя a.
```

Переменные

Переменная – величина, изменяемая при выполнении программы. В тексте присутствует своим именем (уникальный идентификатор).

Тип переменной должен быть объявлен, например:

```
float  x1, y1;  // 4 байта, с плавающей точкой.
int a, b;     // 4 байта, в обратном коде.
char Str [] = "Is string"; // Строка символов.
```

Имена переменных

Имена переменных (идентификаторы), это сочетание букв и цифр, обозначающее в тексте программы объект, изменяющий свое значение.

Имя объекта (переменной) может включать только:

- латинские буквы (большие и маленькие),
- цифры,
- знак подчеркивания.

Первый символ должен быть буквой.

Длина имени <= 31 символу.

Примеры: `x1`, `Price`, `My_file1`, `alpha`, `PI`

Рабочие имена простые: `i`, `j`, `k`.

Ограничения и рекомендации

Не разрешается:

использовать ключевые слова **`for`** **`if`** **`do`**

Не рекомендуется:

использовать имена функций `sin` `sqrt` `ln`

Не следует:

начинать имя со знака _ `_first_name` `_min`

Рекомендации:

Использовать осмысленные имена объектов.

```
float TaxRate;
```

Документировать код с помощью комментариев.

```
float TaxRate; // Налоговый тариф.
```

Механизм переменных

Механизм (семантика) переменных имеет важное значение.

В коде программы программист обязан указать тип любого объекта, например:

```
int a;
```

Компилятор из этого объявления узнает:

1) сколько памяти нужно выделить для размещения переменной, так как размер выделяемой области определен типом переменной, для **int** – 4 байта.

2) Каким образом реализовать внутреннее представление объекта для хранения его значения,

int объекты хранятся в обратном коде.

Концепция типов данных

Тип данного определяет способ хранения данного и представления в оперативной памяти, а именно:

а) размер выделенной памяти (в байтах);

б) представление данного внутри этого адресного пространства (различно для целых, вещественных, символьных и прочих данных);

в) как следствие, множество допустимых значений данного (зависит от объема и способа представления);

г) как следствие, множество операций над данным.

Классификация типов

Самая простая классификация типов, независимо от языка программирования, содержит две группы.

1. Простые типы

   (базовые, скалярные, внутренние, предопределенные)

   встроены в стандарт языка, не могут быть изменены.

2. Конструируемые типы

   нуждаются в построении (описании типа) в соответствии с требованиями задачи.

   Обычно в языке есть инструменты для описания такого типа – массивы, структуры, строки, указатели.

Объявление типа объекта

Тип любого объекта программы должен быть известен обязательно. Для констант тип неявно определен записью. Для переменных обязательно объявление.

Синтаксис объявления объектов:

Тип Имя;

Тип Имя1, Имя2; // Список значений.

Или с инициализацией (присваиванием значения):

Тип Имя = значение;

```
float x1,x2;
float Pi=3.14169236;
const int n=10;
```

В последнем примере объявлена именованная константа.

Основные типы данных

Основные типы данных определены ключевыми словами:

`char` символьный (1 байт – целое значение).

`int` целый (4 байта).

`float`с плавающей точкой (4 байта).

`double` двойной точности (8 байт).

Замечание : для определения размера памяти, выделенной типу данных или объекту используется операция `sizeof()`:

```
sizeof (int)          // Имя типа.
sizeof (My_object)// Имя объекта.
```

Специальный тип **`void`** используется для адресации адресного пространства произвольной длины.

Модификация типа данных

Основные типы могут быть изменены модификаторами.

1. **`long`** – длинный, изменяет тип, удваивая число байт.

   `short` – короткий, принят по умолчанию.

`long double` (16 байт)

`long int` (8 байт)

2. **`unsigned`** – без знака, знаковый разряд передается значению.

   `signed` – со знаком, принят по умолчанию.

Механизм представления: если двухбайтовое **`int`** может принять значение в диапазоне [– 32768 .. + 32767], то **`unsigned int`** в тех же двух байтах может принять значение в диапазоне [0.. 65535].

Определение типа констант

1. Тип константы может быть определен ее записью в коде:

```
1.15   1.5E-12  '!'   "Привет, соня!"
```

2. Можно ввести именованные константы, для чего необходимо объявление, синтаксис которого:

const Имя Тип = значение;

// Значение не может быть изменено при выполнении программы.

```
const float Pi=3.14;
const int A=0;
```

Механизм **define** констант будет рассмотрен ниже.

Таблица типов целых констант

Десятичные	Восмеричные	Шестнадцатиричные	Длинные	Беззнаковые
127	012	0xa	12345l	123u
-256	-014	-0x10	-54321L	123U
0	00	0x0	0l	0u

Суффиксы и префиксы в записи констант порождают требуемое представление, например:

```
0xb80000001    0123LU
```

Действительные константы

Действительные (вещественные) константы имеют тип **`float`** или **`double`**.

Общепринятая запись:

```
1.23        -5.5
```

Научное представление: $M \cdot 10^{P}$

M – мантисса, вещественное число.

P – порядок, вещественное число.

```
1e1=10
1E-1=0.1
-2.34e7=-22400000
9.98*1000000 = 9.98e7
```

Символьные константы (char)

Признак символьной константы – апострофы, например ':' '!' 'S'.

Особые символы – управляющие (ESC-последовательности), начинаются со знака \, например, 'a' '\a'

Некоторые управляющие символы:

'\0' - нулевой символ;

'\n' - новая строка;

'\r' - возврат каретки;

'\f' - новая страница;

'\'' – апостроф;

'\"' - двойная кавычка;

'\\' - обратный слэш;

'\a' - звуковой сигнал.

Логический тип данных

Логический тип данных используется для представления логических значений.

В классическом C его нет, и до сиз пор программисты на C прекрасно обходятся типом **`int`**.

Представление:

значение "Истина" (**`True`**) - все, что отлично от 0,

значение "Ложь" (**`False`**) - значение, равное 0.

В расширении C++ для Visual Studio есть тип **`bool`**.

Переменные в формальном языке

Переменная – объект программы, который изменяет свое значение в процессе выполнения программы.

Любой объект обозначен в тексте программы своим именем. Имя определяет адрес размещения объекта в памяти.

Любой объект имеет тип. Тип определяет объем выделенной памяти и способ хранения переменной.

Объем и способ хранения определяют диапазон возможных значений и операции, разрешенные над объектом.

Роль объявления – выделение памяти для размещения объекта.

Выделение памяти происходит при компиляции программы, это называется механизмом статического распределения памяти.

Существует механизм динамического выделения памяти.

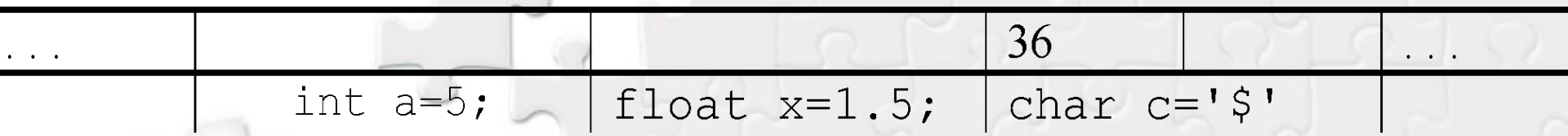
Механизм выделения памяти и адресация

```
int a=5; // 4 байта. Инициализация при объявлении.
float x; // 4 байта.
char c;  // 1 байт.
x=1.5;
c='$';
```

Имя a Адрес 0x002bf818

Имя b Адрес 0x002bf80c

Имя c Адрес 0x002bf803



Операции с переменной

1. Взять значение по указанному адресу.
2. Положить значение по указанному адресу.

Структура и компоненты программы

В Visual Studio программа представлена в виде иерархии объектов.

1. Решение (рабочая область) – совокупность нескольких проектов. Файл решения имеет расширение .sln.

В состав решения могут входить несколько проектов.

2. Проект – объединение нескольких модулей программного кода для совместной компиляции и сборки. Файл проекта имеет расширение .vcproj, и содержит описание модулей проекта и опций проекта.

Программа входит в состав проекта в виде модулей.

Структура и компоненты программы

3. Модуль – файл, содержащий описание данных и алгоритмов для их обработки.

Программа на языке C++ состоит из одного или нескольких модулей – текстовых файлов с расширением "cpp" (**Source** файлы).

Заголовочные файлы с расширениями "h" или "hpp" – **Header** файлы могут входить в проект как включаемые файлы.

Далее см. пример Primer_1.

Директивы препроцессора

Назначение директив – внесение изменений в код программы перед ее компиляцией.

Препроцессор сканирует код программы, находит директивы, и вносит изменения в текст.

Директивы размещаются в модуле программы перед ее кодом.

Директивы начинаются с # и записываются в одну строку.

```
#define
#include
```

Директива #define

Назначение:

а) для задания именованных констант;

б) для задания строк подстановки.

Синтаксис:

```
#define  ИМЯ выражение
```

Механизм действия – макроподстановки. По всему тексту программного кода вместо имени будет подставлено выражение.

Пример.

```
#define SIZE 100
...
int Array [SIZE];  // Везде в коде SIZE заменено 100.
```

Директива `#include`

Назначение – добавление в текст фрагментов программного кода из других файлов.

Синтаксис:

```
#include  <filename>
#include  "filename"
```

Механизм – осуществляются замены в тексте путем добавления текста из файла с именем **`filename`** в точку нахождения директивы **`#include`**.

```
#include <stdio.h>
#include <math.h>
#include "My_function.h"
```

Директива #include

#include <filename> – используется для включения общих заголовочных файлов, осуществляет поиск только в системных каталогах (содержат стандартные заголовки библиотечных функций, объявления которых должны быть известны программе).

Заголовочные файлы (**head**) содержат объявления функций стандартных библиотек.

#include "filename" – используется для включения личных заголовочных файлов, осуществляет поиск файла сначала в текущем каталоге, а затем в системных каталогах (личные файлы, например, тексты функций, констант).

Комментарии

Есть два вида комментариев:

1. Многострочный комментарий записывается в любом месте текста программы в скобках вида

/* */

2. Однострочный комментарий

// Комментирует текущую строку,

// Действует до окончания строки.

Правила документации программного кода.

Пример:

```
int a, b;  // "Это a – число зайцев, b – число белок.
int Rabbit, Squirr;
```