ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ C++

1 лекция

Дисциплина «Программирование микроконтроллеров на языке C++»

Цели и задачи лекции

- Познакомить студентов с понятием языка программирования.
- Рассмотреть классификацию и эволюцию языков программирования.
- Изучить инструментальное программное обеспечение для программирования на C++.

Ключевые вопросы лекции

- Понятие программирования и его роль в автоматизации и управлении.
- Эволюция и классификация языков программирования.
- Инструментальное программное обеспечение.

1. Понятие программирования и его роль в автоматизации и управлении.

Понятие программирования

Программирование – это процесс создания программ, то есть последовательности инструкций, которые выполняет вычислительное устройство (компьютер, микроконтроллер, контроллер промышленной установки и др.) для решения поставленных задач.

Программирование объединяет:

- **алгоритмизацию** (разработка логики решения задачи),
- **формализацию** (запись алгоритма на языке, понятном машине),
- **реализацию** (написание и отладка программы).

Программа выступает связующим звеном между человеком и техническим устройством.

В автоматизации и управлении программирование позволяет:

- **управлять процессами и оборудованием** (например, включение/выключение насосов, регулировка температуры или давления),
- **обрабатывать сигналы от датчиков** (температура, уровень жидкости, скорость вращения),
- **принимать решения на основе логики и алгоритмов** (например, если температура превысила норму — включить систему охлаждения),
- **реализовывать системы человеко-машинного интерфейса** (панели управления, отображение информации для оператора).

Применение микроконтроллеров

Программируемые микроконтроллеры применяются для построения:

- систем управления электроприводами и технологическим оборудованием,
- регуляторов температуры, давления, расхода,
- систем мониторинга и диагностики,
- элементов интеллектуальных систем («умные» сенсоры, системы управления объектами).

Таким образом, программирование является ключевым инструментом инженера в области автоматизации и управления, позволяя реализовывать адаптивные и надёжные системы управления, обеспечивающие требуемую точность и эффективность технологических процессов.

2. Эволюция и классификация языков программирования

На сегодняшний день существует множество языков программирования, которые можно классифицировать по различным критериям.

Основные группы включают:

- машинные языки
- языки ассемблера
- языки высокого уровня
- языки визуального программирования
- интеллектуальные языки программирования

Машинные языки

Машинный код (машинный язык) –система команд (набор кодов операций) конкретной вычислительной машины, которая интерпретируется непосредственно процессором или микропрограммами этой машины.

Особенности:

- состоит из последовательности двоичных цифр (битов) или шестнадцатеричных символов, представляющих инструкции, которые может выполнять процессор.
- машинно-зависимы - специфичен для конкретной архитектуры процессора, что делает код менее переносимым между различными системами.
- обеспечивает высокую скорость выполнения, что важно для задач, требующих высокой производительности.

Пример: (Сложить два числа и результат занести в 3-й адрес)

00101000	00000010	11000000	11000100	11011000
Адрес команды	Код операции	1-й адрес	2-й адрес	3-й адрес

Языки программирования Ассемблер

- **Языки низкого уровня (ассемблеры, автокоды, системные языки)** - текстовые языки, понятные человеку и однозначно переводимые (транслируемые) в языки низкого уровня, то есть машинный двоичный код.

Циклы:

Си:

```
for ( i = 0 ; i < 15 ; i++)
{
    j = j + j;
}
```

Ассемблер:

```
      SUB R0, R0, R0 ; i -> R0 and i = 0
start CMP R0, #15 ; is i < 15?
      ADDLT R1, R1, R1 ; j = j + j
      ADDLT R0, R0, #1 ; i++
      BLT start
```

Ветвления:

Си:

```
if (i == 0)
{
    i = i + 10;
}
```

Ассемблер:

```
@(переменная i находится в регистре R1)
SUBS R1, R1, #0
ADDEQ R1, R1, #10
```

Ассемблер

Ассемблер является основным и наиболее распространённым низкоуровневым языком программирования. Он служит промежуточным звеном между машинным кодом и языками высокого уровня.

Особенности языка ассемблера:

- использование мнемонических кодов вместо двоичных и шестнадцатеричных комбинаций(например, MOV – перемещение данных, ADD – сложение, SUB – вычитание);
- доступ к регистрам процессора и ячейкам памяти, что позволяет точно управлять аппаратными ресурсами;
- возможность применения символьных имён для переменных, констант и адресов памяти, что упрощает написание и чтение кода.

Ассемблер

Ассемблер – это низкоуровневый язык программирования, в котором команды процессора представляются в виде мнемонических обозначений и который предназначен для разработки программ, напрямую взаимодействующих с аппаратным обеспечением компьютера.

Программы на ассемблере транслируются в машинный код, понятный процессору, что обеспечивает полный контроль над вычислительными ресурсами.

Для перевода программ, написанных на ассемблере, в машинный код применяются специальные программы – **трансляторы (ассемблеры)**. Их появление значительно упростило процесс программирования по сравнению с прямым использованием двоичных кодов.

Языки программирования высокого уровня

Языки программирования высокого уровня - это языки, ориентированные на удобство восприятия человеком, использующие синтаксис, приближенный к естественному языку, и позволяющие описывать алгоритмы в абстрактной форме, независимо от архитектуры компьютера.

Программы на языках высокого уровня транслируются в машинный код с помощью компиляторов или интерпретаторов.

Примеры: Python, Java, C++, C#, PHP, JavaScript.

Python

```
d = int(input())
s = 3

while d < 90:
    d += 2
    s += 65

print(s)
```

Pascal

```
var s, d: integer;

begin
   read(d);
   s := 3;

   while d < 90 do begin
      d := d + 2;
      s := s + 65;
   end;
   writeln(s);
end.
```

C++

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

int main()
{
    int s, d;
    cin >> d;
    s = 3;
    while (d < 90)
    {
        d += 2;
        s += 65;
    }
    cout << s;
    return 0;
}
```

Сравнение языков программирования

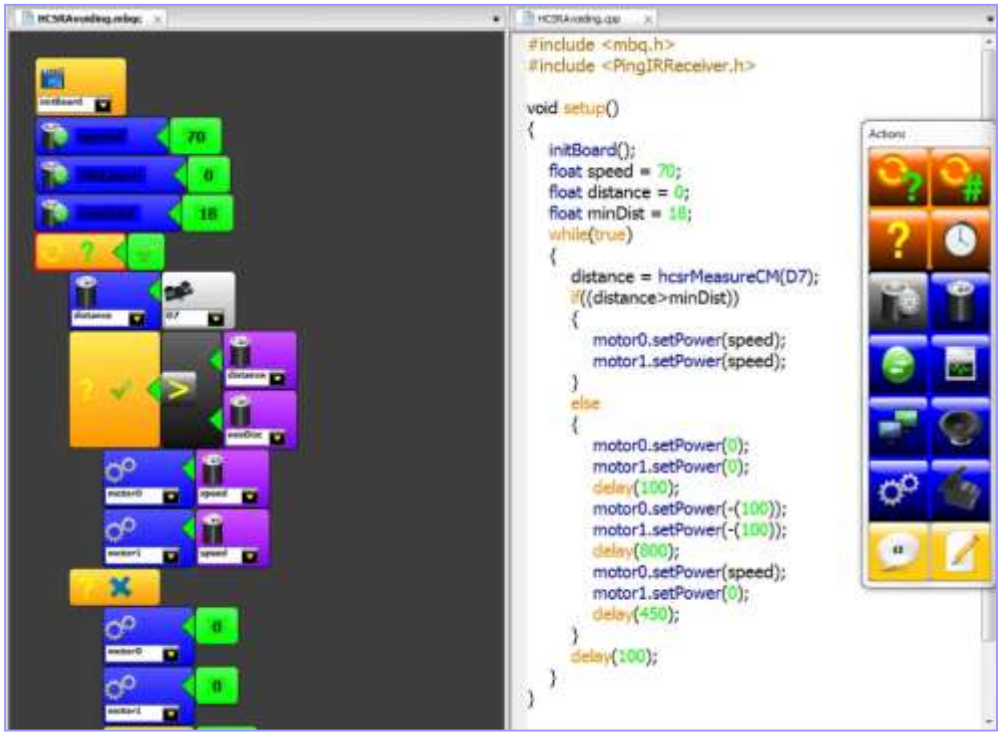
Критерий	Низкоуровневые языки	Высокоуровневые языки
Взаимодействие с аппаратным обеспечением	Максимально приближены к машинному коду, команды напрямую взаимодействуют с архитектурой процессора.	Ориентированы на человека, синтаксис напоминает естественный язык (английский).
Производительность	Позволяют создавать очень быстрый и оптимальный код, эффективный по памяти и ресурсам.	Генерируют менее оптимальный код, который требует больше ресурсов для выполнения.
Требования к преобразованию	Ассемблер переводит инструкции один к одному в машинный код.	Требуют компилятора или интерпретатора для трансляции в машинный код.

Сравнение языков программирования

Критерий	Низкоуровневые языки	Высокоуровневые языки
Портативность	Код зависит от архитектуры конкретного процессора и не переносим между различными системами.	Код переносим, может выполняться на разных устройствах при наличии соответствующей среды выполнения.
Сложность разработки	Высокая: требует глубоких знаний архитектуры ЭВМ и работы с памятью.	Более низкая: ориентированы на удобство программиста, поддерживают абстракции и библиотеки.
Поиск и исправление ошибок	Затруднён, требует много времени.	Упрощён за счёт встроенных инструментов отладки и диагностики.
Примеры языков	Машинный код, Ассемблер.	Python, Java, C++, C#, Pascal.

Языки визуального программирования

Языки визуального программирования - это языки, в которых программы создаются с помощью графических элементов (блоков, схем, визуальных объектов), а не только текстового кода.



Языки визуального программирования

Они активно применяются:

- в образовательных целях для обучения основам алгоритмизации и программирования (например, Scratch, Blockly);
- в специализированных областях проектирования и моделирования (например Tinkercad, App Inventor, LabVIEW).

Характерные особенности:

- использование блоков или визуальных элементов вместо текстового кода;
- упрощение анализа и построения алгоритмов за счёт наглядности;
- снижение порога входа для начинающих программистов;
- высокая производительность разработки, языки программирования четвёртого поколения (4GL).

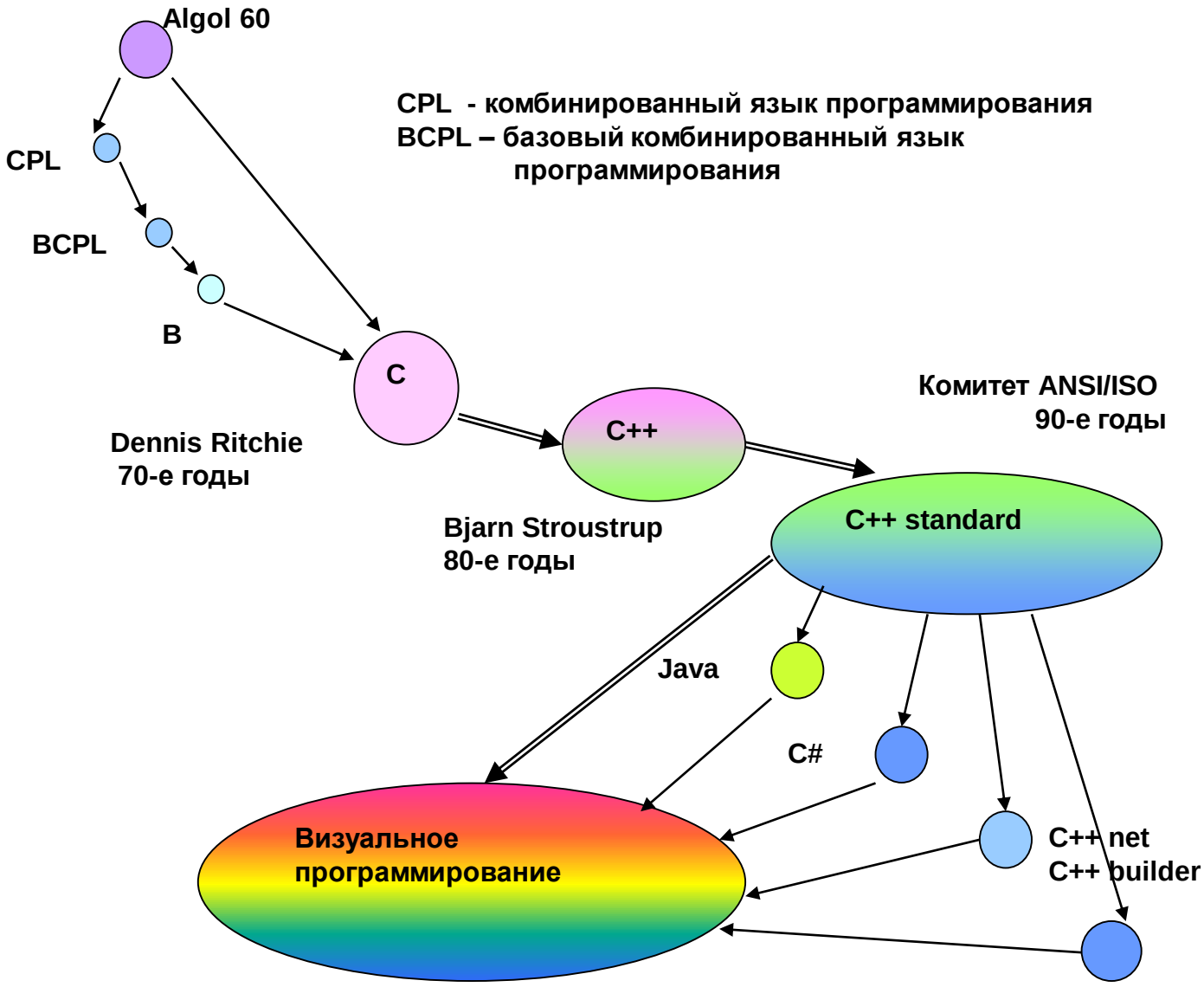
Интеллектуальные языки

Интеллектуальные языки программирования – это языки, в которых человек формулирует задачу на высоком уровне абстракции, а построение алгоритма и его реализация возлагаются на компьютер. Они ориентированы на разработку систем искусственного интеллекта, экспертных систем и баз знаний, используют декларативный и логический подход.

Основные особенности:

- использование логического и декларативного подхода (описание целей и ограничений, а не последовательности действий);
- возможность работы с правилами вывода, фактами и знаниями;
- ориентация на задачи машинного обучения, экспертных систем, обработки естественного языка и интеллектуальных агентов.

Примеры: Prolog, Lisp, OPS5, Mercury.


Algol 60
CPL
BCPL
B
C
Dennis Ritchie
70-е годы
CPL - комбинированный язык программирования
BCPL – базовый комбинированный язык программирования
C++
Bjarn Stroustrup
80-е годы
Комитет ANSI/ISO
90-е годы
C++ standard
Java
C#
Визуальное программирование
C++ net
C++ builder

Языки программирования C и C++: этапы эволюции

- Язык C стал основой для C++. Он был создан в Bell Laboratories в 1969–1973 гг. Кеном Томпсоном и Деннисом Ритчи для разработки ОС UNIX. Предшественниками C были Algol 60, CPL, BCPL и B. Основная цель - получить компактный, эффективный язык, независимый от архитектуры компьютера, но позволяющий напрямую работать с оборудованием.
- Язык C сочетает черты низкоуровневых языков (управление памятью, эффективность) и высокоуровневых (структурность, переносимость). Благодаря компиляторам программы на C можно использовать на разных платформах. Язык стал удобным инструментом для системного программирования и приобрёл статус стандартизованного процедурного языка.

Языки программирования C и C++: этапы эволюции

В 1979 - 1980 гг. в той же фирме Bell Laboratories ее сотрудником Бьерном Страуструпом было разработано расширение языка С, предназначенное для объектно-ориентированного программирования, которое называлось «С с классами».

В 1983 язык был назван C++, что можно (имея в виду операцию инкрементирования языка С) трактовать как увеличенный или расширенный язык С.

В 1989 году Американским Национальным Институтом Стандартизации (ANSI) он был утверждён как ANSI C. ANSI C сейчас поддерживают почти все существующие компиляторы. Любая программа, написанная на стандартном С, гарантированно будет правильно выполняться на любой платформе, имеющей соответствующую реализацию С.

C++ оказал огромное влияние на развитие программирования: его наследниками стали C# и Java, а синтаксис унаследовали PHP, JavaScript и другие языки.

Языки программирования C и C++: этапы эволюции

Существует множество различных реализаций для C++. Каждая из них имеет свои достоинства и недостатки. Наибольшей популярностью пользуются реализации этого языка фирм Borland – Turbo C++, C++ Builder и Microsoft – Visual C++, а также Symantec – Symantec C++.

Язык C++ – это ключ к современному объектно-ориентированному программированию. Он создан для разработки высокопроизводительного программного обеспечения и чрезвычайно популярен среди программистов. Сегодня быть профессиональным программистом высокого класса означает быть компетентным в C++.

C/C++ ценят за его эффективность; он является самым популярным языком для создания системного программного обеспечения (операционных систем, драйверов и др.). Язык C++ – один из основных языков разработки трансляторов.

C/C++ также часто используют для создания прикладных программ. Разработка графического пользовательского интерфейса на языке C++ выполняется, в основном, тогда когда необходимо разрабатывать сложные, нестандартные интерфейсы.

3. Инструментальное программное обеспечение.

Инструментальное программное обеспечение

Инструментальное программное обеспечение обеспечивает создание новых программных продуктов для ПЭВМ. Инструментальное программное обеспечение представлено ***языками*** и ***системами программирования***.

Язык программирования – формальная знаковая система, предназначенная для описания алгоритмов в форме, которая удобна для исполнителя (например, компьютера).

Составными частями языка программирования являются:

- алфавит – конечный набор элементарных символов, разрешенных для использования;
- синтаксис – набор правил образования языковых конструкций и символов алфавита;
- семантика – набор правил однозначного толкования языковых конструкций.

Инструментальное программное обеспечение

Система программирования – это система разработки программного обеспечения на каком-либо языке программирования, включающая в себя следующие компоненты:

- интегрированную среду программирования;
- транслятор;
- редактор текстов программ,
- библиотеки функций,
- программную документацию,
- отладчик и др.

Интегрированная среда программирования (от англ. Integrated Development Environment, IDE) - это программа, имеющая встроенный редактор текстов, подсистему работы с файлами, транслятор, встроенный отладчик, справочную систему, некоторые библиотеки функций. Многие современные среды разработки также включают браузер классов, инспектор объектов и диаграмму иерархии классов для использования при объектно-ориентированном программировании.

Транслятор – программа, которая преобразует команды программ, написанных на языках высокого уровня, в команды, записанные в машинных кодах, использующих двоичный алфавит (в последовательность чисел понятную ПК). Трансляторы бывают двух видов: интерпретаторы и компиляторы. Оба вида трансляторов выполняют одну и туже операцию, но делают это по-разному.

Интерпретаторы, преобразуя команду на машинный язык, сразу же дают указания на ее выполнение, не записывая перевод. Так происходят с каждой командой программы. Программа будет выполнена машиной только при наличие интерпретатора, который от строки к строке переводит команды и сразу же их выполняет. При повторном выполнении программы снова выполняется перевод.

Компилятор же не выполняет команды, а просто переводит всю программу на машинный язык и записывает свой перевод в специальную, так называемую, исполнимую программу, программу, записанную в двоичном коде.

В простейшем случае интерпретатор читает исходный текст программы по одной строке за раз, выполняет эту строку и только после этого переходит к следующей. Компилятор читает сразу всю программу и конвертирует ее в объектный код (машинный двоичный код). Когда программа скомпилирована, в ее коде уже нет отдельных строк исходного кода.

Интегрированные среды программирования

- **Microsoft Visual Studio** - мощная среда разработки с поддержкой C++ и множеством инструментов для разработки.
- **Code::Blocks** - легковесная IDE, поддерживающая различные компиляторы и имеющая настраиваемый интерфейс.
- **Eclipse CDT** - расширение для Eclipse, которое добавляет поддержку C++.
- **CLion** - коммерческая IDE от JetBrains, специально разработанная для C++.
- **Dev-C++** - простая и компактная IDE для разработки на C++.
- **Qt Creator** - IDE для разработки приложений на C++, особенно если вы планируете использовать библиотеку Qt.
- **Xcode** - IDE от Apple для разработки под macOS и iOS, поддерживает C++.
- **NetBeans** - IDE, которая в основном используется для Java, но также поддерживает C++ через плагины.

Консольные и оконные приложения

Консольное приложение – приложение, не имеющее графического интерфейса, для которого устройством ввода является клавиатура, а устройством вывода - монитор.

Консольные приложения удобны как иллюстрации при рассмотрении общих вопросов программирования, когда необходимо сосредоточиться на сути проблемы не задумываясь о пользовательском интерфейсе. На практике консольные приложения применяются для различного вида утилит, тестовых программ и т.д.

Оконное приложение (Windows-приложение) – приложение, в котором используется Windows-интерфейс GUI (Grphical User Interface – графический интерфейс пользователя).

Реализация заданий данного курса будут выполняться в консольных приложениях. GUI-программы отличаются высокой сложностью. Поэтому они не совсем подходят для обучения языку программирования. Освоив C++ на консольных приложениях, вы сможете применить свои знания к Windows-программированию.

Выводы

- В ходе лекции студенты ознакомились с понятием программирования и его значением для автоматизации и управлении.
- Рассмотрены эволюция и классификация языков программирования, показано, какие современные языки унаследовали синтаксис и концепции C++.
- Также студенты познакомились с инструментальным программным обеспечением для разработки на C++, включая компиляторы и среды разработки.
- Лекция заложила базу для понимания принципов программирования и подготовки к практическим занятиям по созданию программ для микроконтроллеров.