**Для чего нужен язык программирования С?**

Создатель С++ Бьярне Страуструп выделяет следующие сферы применения любого языка программирования:

1. Инструмент для инструктирования машин.

2. Способ записи алгоритмов.

3. Средство общения программистов.

4. Инструмент для экспериментирования.

5. Средство управления компьютеризированными устройствами.

6. Способ выражения отношения между концепциями.

7. Средство выражения проектных решений высокого уровня.

Он упоминает, что язык программирования сочетает в себе все указанные свойства,

но главное его предназначение – это создание качественного программного обеспечения.

Каждый язык программирования так или иначе специализирован. Данная специализация была

произведена при создании и стандартизации каждого языка программирования. То есть либо

создатель языка, либо коллектив разработчиков, либо некоторый комитет по стандартизации

определил конкретную предметную область, которой призван служить каждый язык программирования.

Наличие специализации вовсе не предполагает, что некоторый язык программирования не пригоден

ни для чего остального, просто он менее пригоден для этих целей, чем другие специализированные языка программирования.

Язык программирования С обычно называют высокоуровневым аллемблером.

Этот язык разработал Денис Ричи в 1972, будучи сотрудником Bell Labs.

Он является развитием языка B и предназначался для обеспечения удобства разработки

Операционной системы Unix, но со временем его лаконичность и эффективность позволили перенести его на

множество платформ.

Сегодня С используется повсеместно: от приложений для персональных компьютеров для встраиваемых систем и суперкомпьютеров.

Более того, до сих пор распространено мнение, что язык встраиваемых систем - это в первую очередь именно язык С.

Несмотря на развитие и распространение таких довольно популярных потомков как С++, Java, C#, язык С по прежнему

играет важную роль в проектировании и разработки компьютерных программ.

С - это компилируемый высокоуровневый язык общего назначения со строго типизацией и слабой проверкой типов.

Его руководящее предназначение - системное программирование, хотя он хорош и эффективен во всем другом.

Тем не менее программисты считают, "С" слишком сложен в использовании, так как предоставляет мало гарантий

разработчикам приложений, потому для систем, где нет критической нужды в эффективности используются другие высокоуровневые языка

программирования.

Язык С первоначально систематично описан в книге Брайана Кернигана и Дениса Ритчи «Язык программирования Си», которая

на долгое время стала стандартом данного языка.

K&R ввели следующие особенности языка:

1. struct

2. long int

3. unsigned int

4. operator +=

После публикации книги в язык были добавлены такие особенности:

1. Функции типа void

2. Функции, возвращающие структуры

3. Имена полей данных структур в разных пространствах имен для данной структуры

4. Присваивание структур

5. const

6. std lib

7. enum

8. float

Прошел три стандартизации:

1. С89

2. С99

3. С11

Основан на следующих принципах:

1. Простая языковая база, из которой вынесены в библиотеки многие существенные возможности,

вроде математических функций или функций управления файлами;

2. Ориентация на процедурное программирование, обеспечивающая

удобство применения структурного стиля программирования;

3. Система типов, предохраняющая от бессмысленных операций;

4. Использование препроцессора для, например, определения макросов и включения файлов с исходным кодом;

5. Непосредственный доступ к памяти компьютера через использование указателей;

6. Минимальное число ключевых слов;

7. Передачу параметров в функцию по значению, а не по ссылке (при этом передача по ссылке эмулируется с помощью указателей);

8. Указатели на функции и статические переменные;

9. области действия имён;

10.Структуры и объединения — определяемые пользователем собирательные типы данных,

которыми можно манипулировать как одним целым;

11. Средства объектно-ориентированного программирования;

Сила языка С, также как и языка С++ в его пользовательских библиотеках, которые позволили ему вырасти изязыка операционной системы UNIX в высокоуровневый язык общего назначения, при сохранении тех же синтаксических и семантических конструкций.

**Язык «С» - это в первую очередь язык системного программирования.**

Наверное, самое интересное, если не самое парадоксальное в системном программировании на «С» в том, что на «С» создавались и ОС Windows и OS UNIX, но при этом, чтобы создавать приложения, работающие в указанных операционных системах, необходимо использовать библиотеку посредника – Win32API для Windows и POSIX для UNIX, иначе объекты ядра ОС использовать не получится.

К сожалению, «С» не поддерживает парадигму объектно-ориентированного программирования, поэтому создание интерфейса для убогого API Win32 будет выглядеть не настолько приятно, как при использовании классов C++.

**Win32API**

Указанный интерфейс программирования предусматривает собственные типы:

1. BOOL;
2. BYTE;
3. DWORD;
4. INT;
5. LONG;
6. NULL;
7. UINT;

Ядро Windows представляет собой множество ресурсов и множество объектов, которые могут быть использованы при разработке приложений для данной OS (например: окна, файлы, сокеты).

Каждый ресурс и каждый объект – один из множества одновременно используемых, поэтому для идентификации используемого данным приложением объекта было решено использовать специальные дескрипторы:

1. HANDLE – дескриптор объекта.
2. HBITMAP
3. HBPUSH
4. HCURSOR
5. HDC
6. HFONT
7. HICONS
8. HINSTANCE
9. HMENU
10. HPEN
11. HWND

Система данного API предусматривает не только специальные объекты и функции, но и специальные соглашения об именовании. В этом API объекты именуются заглавными буквами, а функции имеют длинные описательные имена. Так, в приведенных выше объектах первая буква H(ANDLE) обозначает, что этот объект является дескриптором соответствующего объекта.

В зависимости от используемых кодировок выделяют объекты используемые Unicode и используемые ANII:

1. LPCSTR – long point constant string
2. LPCTSTR
3. LPCWSTR
4. LPSTR
5. LPTSRT – без Unicode
6. LPWSTR – с Unicode
7. TCHAR

Также Win32Api предусматривает вспомогательные типы:

1. LPARAM;
2. LRESULT;
3. WPARAM;

WinApi32 по сути, это обычная пользовательская библиотека с уникальным системным назначением, только она позволяет обращаться к объектам ядра ОС Windows.

Избавляемся от имен в верхнем регистре

typedef double\_word (DWORD);

typedef byte32 (BYTE);

typedef bool32 (BOOL);

typedef int32 (INT);

typedef unsigned\_int (UINT);

typedef long\_int (LONG);

//дескрипторы

typedef handle (HANDLE);

typedef windows\_handle (HWND);

typedef handle\_curr\_app (HINSTANCE);

//строки

typedef long\_point\_str (LPCSTR);

typedef const\_tstr (LPCTSTR);

typedef const\_wide\_str (LPCWSTR);

//Вводим перечисление типов возможного поведения всплывающих окон:

enum types\_of\_behaviour{

abort\_retry\_ignore = MB\_ABORTRETRYIGNORE,

cancel\_try\_continue = MB\_CANCELTRYCONTINUE,

help = MB\_HELP,

ok = MB\_OK,

ok\_cancel = MB\_OKCANCEL,

retry\_cancel = MB\_RETRYCANCEL,

yes\_no = MB\_YESNO,

yes\_no\_cancel = MB\_YESNOCANCEL

};

// создаем свою функцию BOX

void Box(char\* text, char\* lbl, long type){

MessageBox(NULL,&text[0],&lbl[0],type);

}

int main() {

Box("HEISENBEGR!!!!!!!!","WHAT IS MY NAME???",ok);

}

Это по-прежнему удобнее, чем оригинальное API, но настолько удобнее, чем С++.

Научные вычисления

Выгодное отличие языка Си от С++ является то, что матрицы уже являются частью языка, в том время как в С++ - это только библиотечный класс.

Синтаксис матриц:

int matrix[2][2];

Стандартная библиотека Си поддерживает огромное число функций в том числе:

|  |  |
| --- | --- |
| abs | Возвращает [абсолютную величину](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D0%BE%D0%BB%D1%8E%D1%82%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0) целого числа |
| acos | [арккосинус](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BA%D0%BA%D0%BE%D1%81%D0%B8%D0%BD%D1%83%D1%81) |
| asin | [арксинус](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BA%D1%81%D0%B8%D0%BD%D1%83%D1%81) |
| atan | [арктангенс](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BA%D1%82%D0%B0%D0%BD%D0%B3%D0%B5%D0%BD%D1%81) |
| atan2 | [арктангенс](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BA%D1%82%D0%B0%D0%BD%D0%B3%D0%B5%D0%BD%D1%81) с двумя параметрами |
| ceil | [округление](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D1%80%D1%83%D0%B3%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) до ближайшего большего целого числа |
| cos | [косинус](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%81%D0%B8%D0%BD%D1%83%D1%81) |
| exp | вычисление [экспоненты](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%B0) |
| fabs | [абсолютная величина](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D0%BE%D0%BB%D1%8E%D1%82%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0) (числа с [плавающей точкой](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B2%D0%B0%D1%8E%D1%89%D0%B0%D1%8F_%D1%82%D0%BE%D1%87%D0%BA%D0%B0)) |
| floor | [округление](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D1%80%D1%83%D0%B3%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) до ближайшего меньшего целого числа |
| fmod | вычисление [остатка от деления](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81_%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BA%D0%BE%D0%BC) нацело для чисел с плавающей точкой |
| frexp | разбивает число с плавающей точкой на [мантиссу](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BD%D1%82%D0%B8%D1%81%D1%81%D0%B0) и показатель степени. |
| ldexp | умножение числа с плавающей точкой на целую степень двух |
| log | [натуральный логарифм](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%82%D1%83%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC) |
| log10 | [логарифм по основанию 10](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC#%D0%94%D0%B5%D1%81%D1%8F%D1%82%D0%B8%D1%87%D0%BD%D1%8B%D0%B5_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D1%8B) |
| modf(*x*,*p*) | извлекает целую и дробную части (с учетом знака) из числа с плавающей точкой |
| pow(*x*,*y*) | результат возведения *x* в степень *y*, *xy* |
| sin | [синус](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%83%D1%81) |
| sinh | [гиперболический синус](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D0%B1%D0%BE%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B8%D0%BD%D1%83%D1%81) |
| sqrt | [квадратный корень](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D1%80%D0%B5%D0%BD%D1%8C) |
| tan | [тангенс](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%BD%D0%B3%D0%B5%D0%BD%D1%81) |
| tanh | [гиперболический тангенс](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D0%B1%D0%BE%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D0%B0%D0%BD%D0%B3%D0%B5%D0%BD%D1%81) |

По прежнему, считается, что лидером в программировании встраиваемых систем является как раз язык Си.

Также считается, что язык Си позволят полностью раализовать метод проектирования встраиваемых систем.

Программа на языке Си состоит из основной программы (main.c), которая описывает всю задачу. Основная программа вызывает программы функции, которые выполняют определенные действия. Программы функции можно рассматривать как реализацию модулей, которые мы обсуждали на этапе структурной декомпозиции проекта. Программы функции могут, в свою очередь, вызывать другие функции, в соответствие с иерархической структурой проекта.

Метод проектирования сверху вниз может быть легко реализован на Си. Основная программа показывает общий ход реализации алгоритма управления. Чем дальше Вы погружаетесь в функции все более низкого уровня, тем более детально Вы знакомитесь с особенностями алгоритма управления.

Кроме того, язык СИ позволяет сгенерировать код по быстродействию сравнимый с кодом ассемблера.

Файл листинга обычно используется в процессе отладки прикладной программы при выявлении несоответствий между задуманными программистом действиями и реальным ходом исполнения программного кода. Кроме того, в процессе отладки иногда полезно знать, какие коды должны быть расположены в ячейках с фиксированными адресами. Последнюю информацию наиболее удобно получить из файла карты памяти \*.map (иногда этот файл называют файлом символьных меток). Пример файла карты памяти для программы sample.c приведен ниже.

/\* Название: Sample.c                                       \*/

/\* Описание: Эта программа производит включение             \*/

/\* и выключение светодиодов с интервалом                    \*/

/\* времени 1 с. Используется МК 68НС12ВЗ2                   \*/

/\* Файл заголовка header содержит адреса всех               \*/

/\* портов и регистров специальных функций                   \*/

/\* Контроллер должен быть сконфигурирован                   \*/

/\* для работы в однокристальном режиме                      \*/

/\* Дата создания: May 15, 2004                              \*/

/\* Авторы: Daniel Pack and Steve Barrett                    \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1 #include <68НС12ВЗ2.h>

1 #define \_IO\_BASE 0

2 #define \_P(off) \*(unsigned char volatile\*) (\_IO\_BASE + off)

3 #define TSCR \_Р(0х86)

4 #define TMSK2 \_Р(0х8D)

5 #define TFLG2 \_P(0x8F)

6 #define DDRA \_Р(0х02)

7 #define PORTA \_Р(0х00)

8 #define CLI() asm("cli\n")

9 #define EXIT() asm("swi\n")

2 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

3 void TOIISR(void);

4 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

5 #pragma interrupt\_handler TOIISR() /\* Объявление подпрограммы

6  прерывания по переполнению таймера\*/

7 #pragma abs\_address:0x0B1E /\*задать адрес подпрограммы прерывания ISR \*/

8 void (\* Timer\_Overflow\_interrupt\_vector[])()={TOIISR}

9 #pragma end\_abs\_address

10 unsigned char second = 0x00;

11 void main (void)

12 {

13  TSCR=0x80; /\*включить таймер\*/

14  TMSK2=0x80; /\*разрешить прерывания по таймеру\*/

15  TFLG2=0x80; /\*очистить флаг TOIF\*/

16  DDRA=0xFF; /\* настроить порт Port A на вывод\*/

17  CLI(); /\*разрешить прерывания\*/

19  EXIT()

20 }

21

22 void TOIISR(void) /\*подпрограмма прерывания\*/

23 {

24  TFLG2=0x80; /\*очистить флаг TOIF\*/

25  second += 1; /\*увеличить на 1 программный счетчик с именем second\*/

26  if (second == 122)

27  {

28   PORTA = !PORTA; /\*инвертировать порт PORT A\*/

29   second = 0x00; /\*обнулить программный счетчик\*/

30  }

31 }

Таким образом, можно сделать вывод, что язык Си пусть немного старомоден, но тем не менее остается по прежнему одним и наиболее эффективных языков системного программирования и программирования встраиваемых систем.