Основы алгоритмизации и программирования. Введение в язык программирования Си

Лекция 2

Привезенцев Д.Г.

Муромский институт Владимирского государственного университета
Очная форма обучения

23 сентября 2021 г.

Истоки

- Язык Си был создан во время написания операционной системой UNIX

Истоки

- Язык Си был создан во время написания операционной системой UNIX
- 1969 г. Операционная система писалась на ассемблере, и нужен язык высокого уровня

Истоки

- Язык Си был создан во время написания операционной системой UNIX
- 1969 г. Операционная система писалась на ассемблере, и нужен язык высокого уровня $\rightarrow$ язык B

Истоки

- Язык Си был создан во время написания операционной системой UNIX
- 1969 г. Операционная система писалась на ассемблере, и нужен язык высокого уровня $\rightarrow$ язык B
- 1971 г. Стало ясно, что язык B не совсем подходит для Unix для PDP-11 (мини ЭВМ), тогда стала создаваться расширенная версия языка $\rightarrow$ NB.

Истоки

- Язык Си был создан во время написания операционной системой UNIX
- 1969 г. Операционная система писалась на ассемблере, и нужен язык высокого уровня $\rightarrow$ язык B
- 1971 г. Стало ясно, что язык B не совсем подходит для Unix для PDP-11 (мини ЭВМ), тогда стала создаваться расширенная версия языка $\rightarrow$ NB.
- 1972 г. Язык NB стал сильно отличатся от языка B, поэтому его переименовали в C.

Истоки

- Язык Си был создан во время написания операционной системой UNIX
- 1969 г. Операционная система писалась на ассемблере, и нужен язык высокого уровня $\rightarrow$ язык B
- 1971 г. Стало ясно, что язык B не совсем подходит для Unix для PDP-11 (мини ЭВМ), тогда стала создаваться расширенная версия языка $\rightarrow$ NB.
- 1972 г. Язык NB стал сильно отличатся от языка B, поэтому его переименовали в C.
- 1973 г. На нем переписали операционную систему UNIX.

Истоки

- Язык Си был создан во время написания операционной системой UNIX
- 1969 г. Операционная система писалась на ассемблере, и нужен язык высокого уровня $\rightarrow$ язык B
- 1971 г. Стало ясно, что язык B не совсем подходит для Unix для PDP-11 (мини ЭВМ), тогда стала создаваться расширенная версия языка $\rightarrow$ NB.
- 1972 г. Язык NB стал сильно отличатся от языка B, поэтому его переименовали в C.
- 1973 г. На нем переписали операционную систему UNIX.
- Язык Си продолжил активно развиваться в период с 1977 по 1979.

Истоки

- Язык Си был создан во время написания операционной системой UNIX
- 1969 г. Операционная система писалась на ассемблере, и нужен язык высокого уровня $\rightarrow$ язык B
- 1971 г. Стало ясно, что язык B не совсем подходит для Unix для PDP-11 (мини ЭВМ), тогда стала создаваться расширенная версия языка $\rightarrow$ NB.
- 1972 г. Язык NB стал сильно отличатся от языка B, поэтому его переименовали в C.
- 1973 г. На нем переписали операционную систему UNIX.
- Язык Си продолжил активно развиваться в период с 1977 по 1979.
- Язык подвергся небольшим изменениям в 1995 (изменения описаны в документе, который обычно называют Поправка 1).

Истоки

- Язык Си был создан во время написания операционной системой UNIX
- 1969 г. Операционная система писалась на ассемблере, и нужен язык высокого уровня $\rightarrow$ язык B
- 1971 г. Стало ясно, что язык B не совсем подходит для Unix для PDP-11 (мини ЭВМ), тогда стала создаваться расширенная версия языка $\rightarrow$ NB.
- 1972 г. Язык NB стал сильно отличатся от языка B, поэтому его переименовали в C.
- 1973 г. На нем переписали операционную систему UNIX.
- Язык Си продолжил активно развиваться в период с 1977 по 1979.
- Язык подвергся небольшим изменениям в 1995 (изменения описаны в документе, который обычно называют Поправка 1).
- Более значительные изменения случились в 1999 году.

Истоки

- Язык Си был создан во время написания операционной системой UNIX
- 1969 г. Операционная система писалась на ассемблере, и нужен язык высокого уровня $\rightarrow$ язык B
- 1971 г. Стало ясно, что язык B не совсем подходит для Unix для PDP-11 (мини ЭВМ), тогда стала создаваться расширенная версия языка $\rightarrow$ NB.
- 1972 г. Язык NB стал сильно отличатся от языка B, поэтому его переименовали в C.
- 1973 г. На нем переписали операционную систему UNIX.
- Язык Си продолжил активно развиваться в период с 1977 по 1979.
- Язык подвергся небольшим изменениям в 1995 (изменения описаны в документе, который обычно называют Поправка 1).
- Более значительные изменения случились в 1999 году.
- В 2011 году вместе с редакцией языка Си++ был выпущен стандарт C11.

Простейшая программа на Си

```c
#include <stdio.h>

int main() {
  printf("Hello, World!");
  getchar();
  return 0;
}
```

Переменные

Переменные – поименованные данные, которые могут изменяться в процессе выполнения программы.

Объявление переменной — это оператор языка Си, который выглядит следующим образом:

тип идентификатор[=значение];

Тип задается соответствующим ключевым словом, например, *int* или *char*.

Тип данных определяет формат представления переменной в компьютере и множество операций, которые могут выполняться над этой переменной.

Идентификатор — в данном случае это имя переменной.

```
int i, y;
double x, y;
```

Инициализация переменных

Переменные при объявлении могут быть инициализированы.
Инициализация переменной — присвоение начального значения.
Для инициализации переменной необходимо при объявлении переменной определить значение.

```
int s=56;
```

Типы переменных

Тип переменной определяет:

Типы переменных

Тип переменной определяет:

1. Размер переменной в байтах (сколько байт памяти выделит компьютер для хранения значения)

Типы переменных

Тип переменной определяет:

1. Размер переменной в байтах (сколько байт памяти выделит компьютер для хранения значения)
2. Представление переменной в памяти (как в двоичном виде будут расположены биты в выделенной области памяти).

Типы переменных

Тип переменной определяет:

1. Размер переменной в байтах (сколько байт памяти выделит компьютер для хранения значения)
2. Представление переменной в памяти (как в двоичном виде будут расположены биты в выделенной области памяти).

В си несколько основных типов. Разделим их на две группы - целые и числа с плавающей точкой.

Здесь следует сделать замечание.

Размер переменных в си не определён явно, как размер в байтах. В стандарте только указано, что

char <= short <= int <= long <= long long

Целые типы данных

- char - размер 1 байт.

Здесь следует сделать замечание.

Размер переменных в си не определён явно, как размер в байтах. В стандарте только указано, что

char <= short <= int <= long <= long long

Целые типы данных

- char - размер 1 байт.
- short - размер 2 байта

Здесь следует сделать замечание.

Размер переменных в си не определён явно, как размер в байтах. В стандарте только указано, что

char <= short <= int <= long <= long long

Целые типы данных

- char - размер 1 байт.
- short - размер 2 байта
- int - размер 4 байта

Здесь следует сделать замечание.

Размер переменных в си не определён явно, как размер в байтах. В стандарте только указано, что

char <= short <= int <= long <= long long

Целые типы данных

- char - размер 1 байт.
- short - размер 2 байта
- int - размер 4 байта
- long - размер 4 байта

Здесь следует сделать замечание.

Размер переменных в си не определён явно, как размер в байтах. В стандарте только указано, что

char <= short <= int <= long <= long long

Целые типы данных

- char - размер 1 байт.
- short - размер 2 байта
- int - размер 4 байта
- long - размер 4 байта
- long long - размер 8 байт.

Здесь следует сделать замечание.

Размер переменных в си не определён явно, как размер в байтах. В стандарте только указано, что

char <= short <= int <= long <= long long

Целые типы данных

Тип	Размер, байт	Минимальное значение	Максимальное значение
unsigned char	1	0	255
signed char (char)	1	-128	127
unsigned short	2	0	65535
signed short (short)	2	-32768	32767
unsigned int (unsigned)	4	0	4294967296
signed int (int)	4	-2147483648	2147483647
unsigned long	4	0	4294967296
signed long (long)	4	-2147483648	2147483647
unsigned long long	8	0	18446744073709551615
signed long long (long long)	8	-9223372036854775808	9223372036854775807

sizeof

```
1   #include<stdio.h>
2   int main() {
3     char c;
4     short s;
5     int i;
6     long l;
7     long long L;
8
9     printf("sizeof(char)  = %d\n", sizeof(c));
10    printf("sizeof(short) = %d\n", sizeof(s));
11    printf("sizeof(int)   = %d\n", sizeof(i));
12    printf("sizeof(long)  = %d\n", sizeof(l));
13    printf("sizeof(long long) = %d\n", sizeof(L));
14    getchar();
15  }
```

Здесь следует сделать замечание.

Здесь также приведены значения для VC2012, по стандарту размер типов **float** $<=$ **long float** $<=$ **double** $<=$ **long double** все числа с плавающей точкой - со знаком.

Типы с плавающей точкой

- float - 4 байта,

Здесь следует сделать замечание.

Здесь также приведены значения для VC2012, по стандарту размер типов **float** $<=$ **long float** $<=$ **double** $<=$ **long double** все числа с плавающей точкой - со знаком.

Типы с плавающей точкой

- float - 4 байта,
- long float - 8 байт

Здесь следует сделать замечание.

Здесь также приведены значения для VC2012, по стандарту размер типов **float** $<=$ **long float** $<=$ **double** $<=$ **long double** все числа с плавающей точкой - со знаком.

Типы с плавающей точкой

- float - 4 байта,
- long float - 8 байт
- double - 8 байт

Здесь следует сделать замечание.

Здесь также приведены значения для VC2012, по стандарту размер типов **float** $<=$ **long float** $<=$ **double** $<=$ **long double** все числа с плавающей точкой - со знаком.

Типы с плавающей точкой

- float - 4 байта,
- long float - 8 байт
- double - 8 байт
- long double - 8 байт.

Здесь следует сделать замечание.

Здесь также приведены значения для VC2012, по стандарту размер типов **float** <= **long float** <= **double** <= **long double** все числа с плавающей точкой - со знаком.

Переполнение переменных

```
1   #include <stdio.h>
2
3   void main() {
4     unsigned a = 4294967295;
5     int b = 2147483647;
6
7     printf("%u\n", a);
8     a += 1;
9     printf("%u\n", a);
10
11    printf("%d\n", b);
12    b += 1;
13    printf("%d\n", b);
14    getchar();
15  }
```

Постфиксное обозначение типа

При работе с числами можно с помощью литер в конце числа явно указывать его тип, например:

Постфиксное обозначение типа

При работе с числами можно с помощью литер в конце числа явно указывать его тип, например:

- 11 - число типа int

Постфиксное обозначение типа

При работе с числами можно с помощью литер в конце числа явно указывать его тип, например:

- 11 - число типа int
- 10u - unsigned

Постфиксное обозначение типа

При работе с числами можно с помощью литер в конце числа явно указывать его тип, например:

- 11 - число типа int
- 10u - unsigned
- 22l или 22L - long

Постфиксное обозначение типа

При работе с числами можно с помощью литер в конце числа явно указывать его тип, например:

- 11 - число типа int
- 10u - unsigned
- 22l или 22L - long
- 3890ll или 3890LL - long long (а также lL или Ll)

Постфиксное обозначение типа

При работе с числами можно с помощью литер в конце числа явно указывать его тип, например:

- 11 - число типа int
- 10u - unsigned
- 22l или 22L - long
- 3890ll или 3890LL - long long (а также lL или Ll)
- 80.0f или 80.f или 80.0F - float

Постфиксное обозначение типа

При работе с числами можно с помощью литер в конце числа явно указывать его тип, например:

- 11 - число типа int
- 10u - unsigned
- 22l или 22L - long
- 3890ll или 3890LL - long long (а также lL или Ll)
- 80.0f или 80.f или 80.0F - float
- 3.0 - число типа double

Постфиксное обозначение типа

```
1  #include<stdio.h>
2  int main() {
3
4    printf("sizeof(int) = %d\n", sizeof(10));
5    printf("sizeof(unigned) = %d\n", sizeof(10u));
6    printf("sizeof(long) = %d\n", sizeof(10l));
7    printf("sizeof(long long) = %d\n", sizeof(10ll));
8    printf("sizeof(float) = %d\n", sizeof(10.f));
9    printf("sizeof(double) = %d\n", sizeof(10.));
10   printf("sizeof(double) = %d\n", sizeof(10e2));
11
12   getchar();
13 }
```

Объявление переменных

При объявлении переменной пишется её тип и имя.

```
int a;
double parameter;
```

Можно объявить несколько переменных одного типа, разделив имена запятой

```
long long arg1, arg2, arg3;
```

Объявление переменных

```
1   #include <stdio.h>
2
3   int main() {
4     int a = 10;
5     int b;
6     while (a>0){
7       int z = a*a;
8       b += z;
9     }
10    getchar();
11  }
```

Начальное значение переменной

Переменные в си НЕ инициализируются по умолчанию нулями, как во многих других языках программирования.
После объявления переменной в ней хранится *мусор* - случайное значение, которое осталось в той области памяти, которая была выделена под переменную.

```
#include<stdio.h>

int main() {
  int i;
  printf("%d", i);
  getchar();
}
```

Область видимости переменной

Переменные бывают *локальными* (объявленными внутри какой-нибудь функции) и *глобальными*.

Глобальная переменная видна всем функциям, объявленным в данном файле.

Локальная переменная ограничена своей областью видимости. Когда говориться, что переменная "видна в каком-то месте это означает, что в этом месте она определена и её можно использовать.

Область видимости переменной. Пример 1

```
1   #include<stdio.h>
2   int global = 100;
3   void foo() {
4     printf("foo: %d\n", global);
5   }
6   void bar(int global) {
7     printf("bar: %d\n", global);
8   }
9   int main() {
10    foo();
11    bar(333);
12    getchar();
13  }
```

Область видимости переменной. Пример 2

```
1 #include<stdio.h>
2
3 int global = 100;
4
5 int main() {
6   int global = 555;
7   printf("%d\n", global);
8   getchar();
9 }
```

Область видимости переменной. Пример 3

```
1   #include<stdio.h>
2
3   int global = 100;
4
5   int main() {
6     int x = 10;
7     {
8       int y = 30;
9       printf("%d", x);
10    }
11    printf("%d", y);
12    getchar();
13  }
```

Область видимости переменной. Пример 4

```
1
2   #include<stdio.h>
3
4   int global = 100;
5
6   int main() {
7     int x = 10;
8     {
9       int x = 20;
10      {
11        int x = 30;
12        printf("%d\n", x);
13      }
14      printf("%d\n", x);
15    }
16    printf("%d\n", x);
17    getchar();
18  }
```