Основы алгоритмизации и программирования.
Часть 2. Система программирования на языке Си.

Для специальностей

1-40 01 01 «Программное обеспечение информационных технологий»

1-40 05 01 «Информационные системы и технологии»

Макареня Сергей Николаевич,
кандидат технических наук, доцент
E-mail: makarenya@bntu.by
Скайп: makar_sn
Тел. +375 (29) 562-82-95

- Курсы 1
- Семестры 2 Курсовая работа, Экзамен

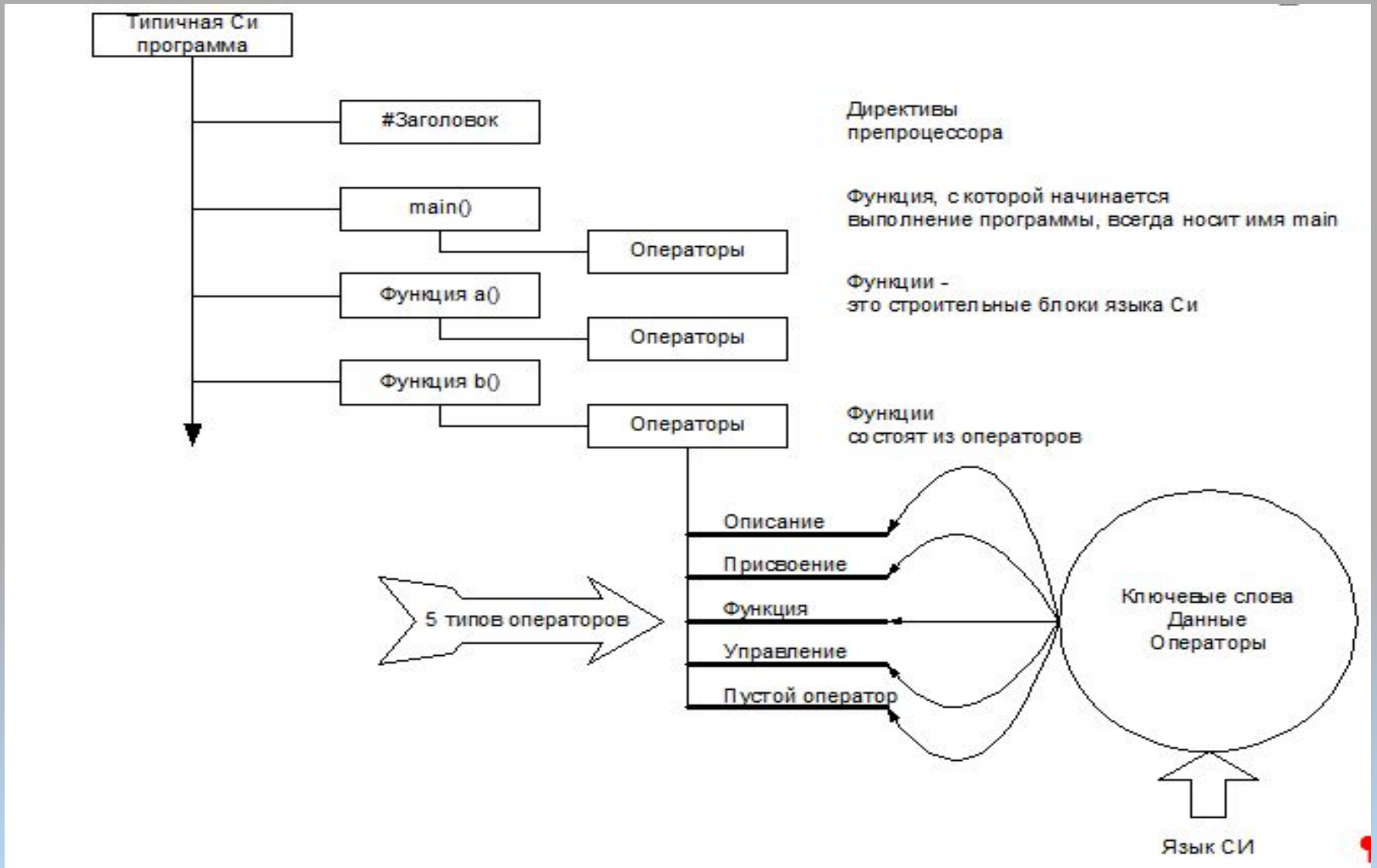
Введение в язык Си

Язык Си создал Деннис Ритчи в 1972 году.

Достоинства языка Си:

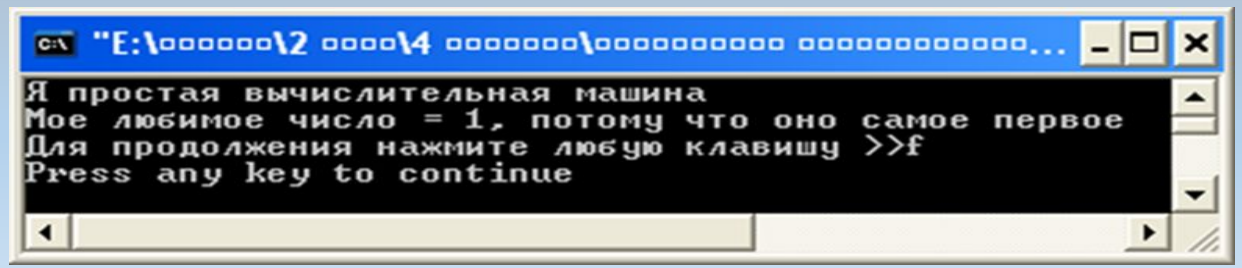
- Структура языка Си побуждает программиста использовать в своей работе нисходящее проектирование, структурное программирование и пошаговую разработку модулей. Результатом такого подхода является надежная и читаемая программа.
- Си — эффективный язык. На языке Си программы обычно отличаются компактностью и быстротой исполнения.
- Си — переносимый, или мобильный, язык.
- Си — мощный и гибкий язык. Программы, написанные на Си, используются для решения физических и технических проблем.
- Си — удобный язык. Он достаточно структурирован, чтобы поддерживать хороший стиль программирования, и вместе с тем не связывать вас смирительной рубашкой ограничений.

Структура программы, написанной на языке Си.



Пример простой программы на языке Си

```
 /*Программа будет сохранена в файле prim1.c */
#include<stdio.h>
void main(void) /*Первая программа*/
{
int n;
n = 1;
printf("Я простая");
printf("вычислительная машина\n");
printf("Мое любимое число = %d, потому что оно самое первое\n",n);
/* Это комментарий, который не обрабатывается программой */
printf("Для продолжения нажмите любую клавишу >>");
scanf ("%d",&n);
}
```



Данные

Данные – информация, преобразованная к виду, пригодному для обработки ЭВМ (ГОСТ).

Переменные:

символьные , целые, вещественные

Константы:

Тип()	Размер памяти в байтах	Диапазон значений
char	1	от -128 до 127
int	2-4	
short	2	от -32768 до 32767
long	4	от -2 147 483 648 до 2 147 483 647
unsigned shar	1	от 0 до 255
unsigned int	2-4	
unsigned short	2	от 0 до 65535
unsigned long	4	от 0 до 4 294 967 295
float	4(32=1зн+8 пор+23 ман.)	3.14E-38 до 3.14E+38.
double	4(64=1зн+11 пор+52 ман.)	1.7E-308 до 1.7E+308.

Примеры объявления данных

Формат объявления переменной: **Тип Переменной Имя_Переменной**;

int n,m; // объявлены переменные n и m целого типа

long z; // объявлена переменная z целого типа, размером 4 байта

unsigned int a, b; // объявлены беззнаковые переменные a и b целого типа

float c; // объявлена переменная с вещественного типа

Общий вид определения переменной:

Тип Переменной Имя_Переменной=Значение;

int x=7, y=41; float r=3.6, y=2.4e-2

Общий вид объявления константы:

const Тип Имя_Константы=Значение;

const float PI=3.14; // объявляется переменная-константа PI,
// которой присваивается значением 3.14

const double A = 2.128E-2;

const B = 286; (подразумевается const int B = 286)

Ввод-вывод данных

Под функциями ввода-вывода подразумеваются функции, которые выполняют транспортировку данных в программу и из нее.. В языке Си осуществлять ввод-вывод можно двумя способами:

- потоковый ввод/вывод данных;
- форматированный ввод/вывод данных.

Функция **getchar()** получает один символ, поступающий с пульта терминала и передает его выполняющейся в данный момент программе.

Функция **putchar()** получает один символ, поступающий из программы, и пересылает его для вывода на экран.

```
 char a,ch;
putchar ('S');       /* напомним, что символьные */
putchar ('\n');      /* константы заключаются в апострофы */
putchar (ch);      /* ch — переменная типа char */
putchar (getchar ())
a=getchar ()
putchar (a);
```

Форматированный ввод-вывод данных

Для вывода используется функция **printf(),** а для ввода **scanf().**

printf(“управляющая строка”, [список вывода]);

Управляющая строка содержит два типа объектов:

- Простые символы, которые копируются в выходной поток;
- Спецификаторы формата, которые применяются к элементам списка вывода.

Формат	**Тип выводимой информации**
%d	Десятичное целое число
%c	Один символ
%s	Строка символов
%e	Вещественное число с плавающей точкой, экспоненциальная запись
%f	Вещественное число , десятичная запись

Пример функции printf()

```
#include <stdio.h>
int main()
{
char ch = 'A';
int z = -1234;
double fp =-123.4567;
//Отображение целых
printf("Вывод целых: \n %d    %.6d \n",z, z, );
// Отображение символов
printf("Символы:\n%c  \n",ch);
//Отображение чисел с плавающей точкой
printf("Вещественные числа:\n%f  %.2f \n%e %9.2e\n", fp, fp,
fp, fp);
return 0;
}
```

```
Вывод целых:
-1234 -001234
Символы:
A
Вещественные числа:
-123.456700 -123.46
-1.234567e+02 -1.23E+02
```

Пример функции scanf()

scanf() (“управляющая строка”, список ввода);

scanf(“%d%f”, &x, &y);

```
void main()
{ int a;
float b,c;
char ch=’A’;
char string[]=”BNTU”;
printf(“%d\n”, 455);                  // вывод на экран целого числа
printf(“%f\n”, 12.34);                // вещественного числа
printf(“%c%s\n”, ch, string);   // вывод символа и строки
printf(“Введите целую переменную:\n”);
scanf(“%d”, &a);                      // ввод целой переменной a
printf(“%d\n”, a);                   // вывод переменной a
printf(“Введите вещественные переменные:\n”);
scanf(“%f%f”, &b, &c);          // ввод вещественных  b и c
printf(“%f%f”, b, c);          // вывод на экран значений  b и c
}
```

```
455
12.340000
A BNTU
Введите целую перемен:
1000
1000
Введите вещественные
переменные:
1.1 2.2
1.1  2.2
```

Программирование разветвляющихся вычислительных процессов

Оператор выбора **if** и оператор с множественным выбором **switch**.

Оператор if имеет следующий формат:

if (выражение) оператор 1; [else оператор 2;]

Оператор if работает следующим образом:

1. Вычисляется значение выражения.
2. Если значение выражение истинно, то выполняется оператор 1, а если ложно, то выполняется оператор 2, следующий за ключевым словом **else**. Операторы могут быть простыми или составными (Составной оператор включает несколько простых операторов, заключенных в фигурные скобки).

Программирование разветвляющихся вычислительных процессов

```
#include<stdio.h>
void main()
{ int n;
float y, x=3.55;
printf (‘введите  значение степени 2 или 3 \n’);
scanf(‘%d\n’,&n);
if (n==2) y=x*x;
else y=x*x*x;
printf(“У = %6.3f ”, y);
}
```

```
введите  значение степени 2 или 3
2
Y= 12.602
```

Множественный выбор: конструкция else-if

В одном операторе можно использовать столько конструкций else-if, сколько нужно, что иллюстрируется приведенным ниже фрагментом:

```
if (score < 1000) bonus = 0;
else if (score < 1500) bonus = 1;
else if (score < 2000) bonus = 2;
else if (score < 2500)  bonus = 4;
else bonus=6;
```

Существует правило, которое гласит, что **else** соответствует ближайшему **if**, кроме случаев, когда имеются фигурные скобки.

Операции отношения и выражения.

Операции отношения используются для сравнений. Мы уже использовали ранее некоторые из них, а сейчас приведем полный список операций отношения, применяемых при программировании на языке Си.

Операция	Смысл
<	меньше
< =	меньше или равно
==	**равно**
>	больше
=	не равно

Главное предостережение, которое необходимо сделать, состоит в том, чтобы не использовать знак **«=»** вместо **«==»**.

(s = 3 присваивает значение 3 переменной canoes
s == 5 проверяет, равняется ли значение переменной s значению 5)

Операция условия: ?:

M=Выражение 1**?** Выражение 2 **:** выражение 3

1. Вычисляется выражение 1.
2. Если выражение1 истинно (больше нуля), то переменной M присваивается значение выражения 2; если выражение 1 ложно (равно 0), то переменной M присваивается значение выражения 3.

x = (y < 0) ? -y : y;	**if (y<0) x= -y;** **else x=y;**

Использование условных выражений не является обязательным, поскольку тех же результатов можно достичь при помощи операторов if-else. Однако условные выражения более компактны, и их применение обычно приводит к получению более компактного машинного кода.

Множественный выбор: операторы switch и break

Оператор switch имеет следующий формат:

switch (выражение)
{ case метка1: оператор1; break;
case метка2: оператор2; break;
• …
case меткаN: операторN; break;
default: операторN+1; break; }

Алгоритм работы оператора switch:

1. Вычисляется выражение в скобках.
2. Сравнивается полученное значение с константами, представленными в метках.
3. С случае, совпадения значений вычисляется оператор соответствующей метки. При наличии оператора break, происходит выход из оператора switch.
4. Если совпадений нет, то управление передается оператору, стоящему после ключевого слова default.

Пример применения оператора множественного выбор

```
#include<stdio.h>
void main()
{
int nd;                    // дни недели
printf(“Введите номер дня недели (1…7):”);
scanf(“%d”,&nd);
switch(nd)                 // структура со множественным выбором
{
case 1: puts(“Понедельник”); break;
case 2: puts(“Вторник”); break;
case 3: puts(“Среда”); break;
case 4: puts(“Четверг”); break;
case 5: puts(“Пятница”); break;
case 6: puts(“Суббота”); break;
case 7: puts(“Воскресение”); break;
default: puts(“Число должно быть в диапазоне от 1 до 7”); break;
}}
```

Результаты работы программы:

Введите номер дня недели: 1
Понедельник

Программирование циклических вычислительных процессов

В языке Си существуют удобные операторы организации цикла: **while, do-while, for.**

Оператор цикла с предусловием **while** .

Формат оператора:

while(выражения)

оператор;

Оператор (тело цикла) - простой или составной оператор.

Выражение – условие выхода из цикла.

При построении цикла **while** в теле цикла должны быть включены конструкции, изменяющие величину проверяемого выражения так, чтобы, в конце концов, оно стало ложным. В противном случае выполнение цикла никогда не завершится.

Пример использования оператора while

```
#include<stdio.h>
void main()
{
int a=0, b=1, n;
puts("Введите количество чисел:");
scanf("%d", &n);
while(a!=n)                    //  использование структуры повторения while
{ if(b%2==0)                   // пока значение условия (a!=n) является истина
{ printf("%d\n", b);           //  выполняется тело цикла
a++; }
b++;
}}
```

Результаты работы программы:

```
Введите количество чисел: 5
2
4
6
8
10
```

Оператор цикла do-while

Оператор **do -while** называется циклом с постусловием и отличается от цикла **while** тем, что сначала выполняется тело цикла, а затем проверяется условие выхода из цикла.

Поскольку циклы **do-while** выполняются, по меньшей мере, один раз, их лучше использовать тогда, когда нет сомнений о вхождении в определенный цикл. Например, если программа должна выводить пользовательское меню, то даже если пользователь захочет тут же выйти из программы, он должен увидеть это меню, чтобы определить, по какой клавише выходить из приложения.

```
do {
    действие1;
    действие2;
    действие3;
    действиеn;
} while(проверка условия выхода из цикла) ;
```

Пример использования оператора do-while

```
#include<stdio.h>
void main()
{
int a;
int m; puts(“Введите числа. Для завершения ведите 0:\n”);
m=0;
do {scanf(“%d”,&a);
if(a>m) m=a;
} while(a>0);
printf(“Максимальное число : %d”, m);
}
```

Результат выполнения программы:

Введите числа. Для завершения ведите 0:

5 3 7 9 0

Максимальное число: 9

Оператор цикла for

Структура повторения for используется для организации циклов с фиксированным, известным во время разработки программы, числом повторений.

Общий вид структуры повторения for:

for (инициализация управляющей переменной; условие; изменение управляющей переменной)

{оператор1; оператор2; оператор3;}

Или

for(i=0;i<10;i=i+1)

{Тело цикла}

Примеры использования оператора for

#include <stdio.h> void main() { int res=0; for(int i=0;i<10;i++) // использование структуры повторения for // объявление переменной i при //инициализации res+=i; // цикл повторяется десять раз printf(“ Сумма равна%4d”,rez); }	**Сумма равна 45**

Использование **for** для реализации в программе временной задержки с целью согласования скорости реагирования машины с возможностями восприятия человека.

- **for(n = 1; n <= 10000; n++)**
- **;**

Примеры использования оператора for

Применение **for** с убыванием значения переменной цикла вместо возрастания.

for (n = 10; n > 0; n--) **printf(" %d секунд!\n", n);** **printf(" Пуск!\n");**	**10** **9** **…** **1** **Пуск**

Произвольный шаг изменения переменной цикла:

```
for (n = 2; n < 60; n = n + 13)
printf(" %d\n" , n);
```

Шаг изменяется в геометрической, а не в арифметической прогрессии:

```
for (debt = 100.0; debt < 150.0; debt = debt* 1.1)
printf(" Ваш долг теперь $%.2f.\n" , debt);
```

Примеры использования оператора for

Третье выражение - любое правильно составленное выражение.

```
for (x = 1; y <= 75; y = 5*x++ + 10)
printf("%10d %10d\n", x, y);
```

Параметры, входящие в выражения, находящиеся в спецификации цикла, можно изменить при выполнении операций в теле цикла. Предположим, например, что у вас есть цикл со спецификацией следующего вида:

```
delta=3;
for(n = 1; n < 1000; n += delta)
```

- И, если после нескольких итераций, ваша программа решает, что величина параметра **delta** слишком мала или велика, оператор if внутри цикла может изменить значение параметра. В диалоговой программе пользователь может изменить этот параметр в процессе выполнения цикла.

Операторы безусловной передачи управления

Выполнение оператора break приводит к выходу цикла и переходу к следующему оператору программы

```
#include<stdio.h>
main()
{ int x;
printf("Вывод значений x\n");
for(x=1;x<=10;x++)
{ if(x==5) break;        // break прерывает цикл
                         //  когда x станет равным 5
printf("X : %2d",x);}
return 0;
}
```

Вывод значений X
X: 1 2 3 4

Операторы безусловной передачи управления

Оператор **continue** вызывает пропуск «оставшейся» части итерации и переход к началу следующей.

Оператор **goto** считается «чрезвычайно плохим» средством и предлагают «применять его как можно реже или не применять совсем».

goto part2.

Чтобы этот оператор выполнился правильно, необходимо наличие другого оператора, имеющего метку part2; в этом случае запись оператора начинается с метки, за которой следует двоеточие:

part2: printf(" Уточненный анализ:\n").

Существует один случай, когда использование оператора **goto** допускается опытными программистами, работающими на языке Си,— это выход из вложенного набора циклов при обнаружении каких-то ошибок.