



ОСНОВЫ программирования

Введение



Данные и алгоритмы

Алгоритм + Структура данных = Программа

Информация

- Информация – уменьшение степени неопределенности знания
- Информация и содержание
- Сопоставление информации реальным объектам называется кодировкой
- Информация – дискретна

Единицы измерения информации

- Минимальная единица – 1 бит
(два дискретных состояния)
- 1 байт – 8 бит
(256 состояний, использовался для представления одного символа)
- 1 слово – 2-4-8 байт, в зависимости от компьютера
- Производные единицы
 - 1К = 1024 байта
 - 1М = 1024К = 1 048 576 байт
 - 1Г = 1024М = 1 073 741 824 байт

Компьютерные модели данных

- Целые числа
(8-16-32-64 бита, знаковые/беззнаковые)
- Адреса
(обычно 32 бита, те же целые числа)
- Символы
(8-16 бит, те же целые числа)
- Вещественные числа
(32-64-80 бит, мантисса+экспонента)

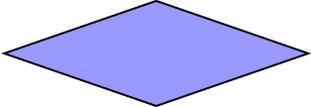
Структуры данных

- Скалярные типы данных
 - Целочисленные
 - Вещественные
 - Символьные
 - Логические
- Массивы
- Структуры
- Объекты

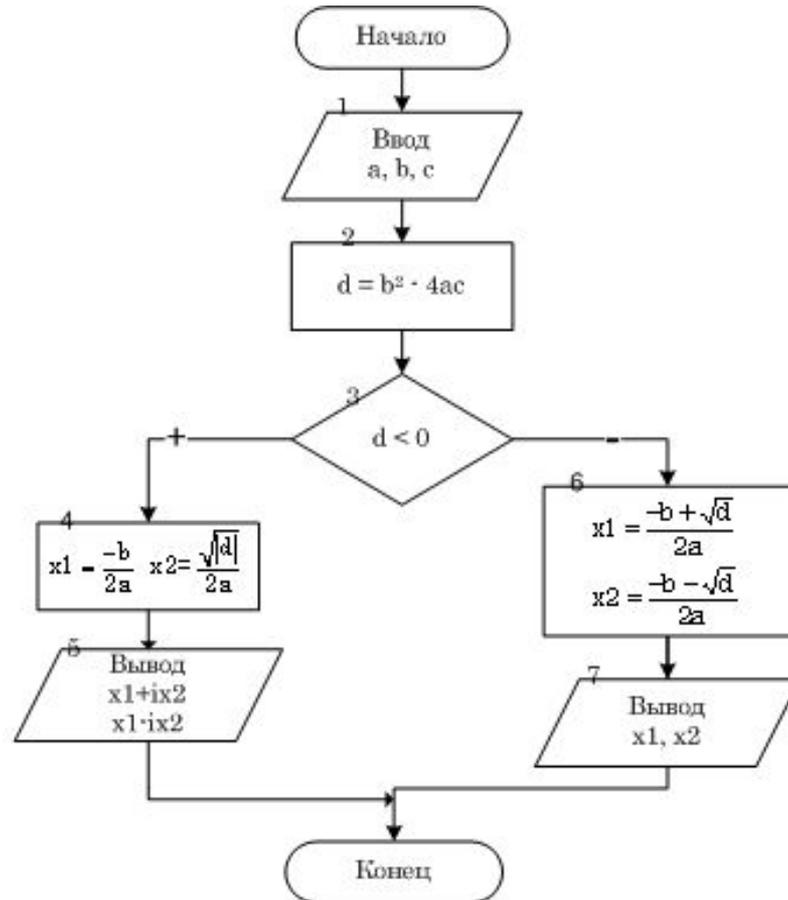
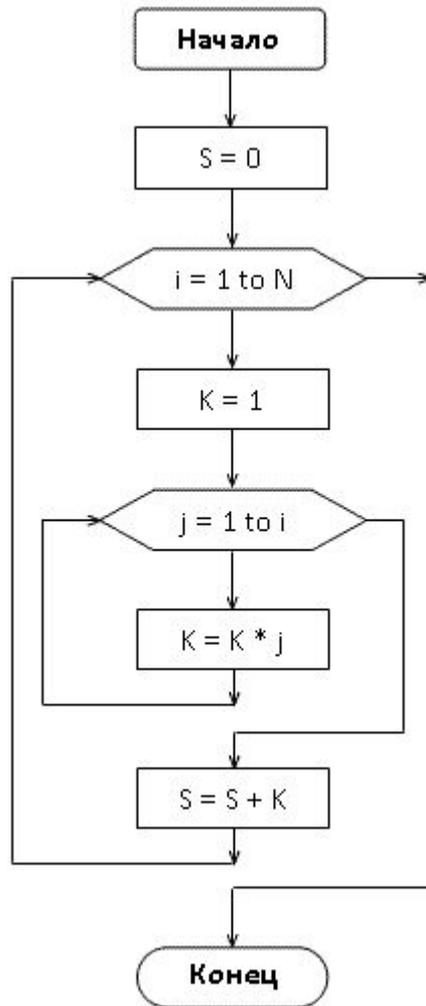
Алгоритм

- Последовательность действий
 - Конечная
 - Однозначно трактуемая
- Описание алгоритма
 - На естественном языке
 - Графическое (блок-схемы)
 - Алгоритмический язык

Блок-схемы

| | |
|---|--------------------|
|  | Вход/выход |
|  | Действие |
|  | Ввод/вывод |
|  | Условие, цикл |
|  | Вызов подпрограммы |
|  | Цикл с параметром |

Пример блок-схем



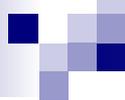
Языки программирования

■ Универсальные

- C/C++
- C#
- FORTRAN
- PASCAL
- BASIC
- LISP
- PROLOG

■ Специализированные

- HTML
- JavaScript
- SQL
- UNIX-shell
- VBA
- VBScript
- PHP



Алгоритмические языки

- C/C++
- C#
- Pascal/Delphi
- Visual Basic
- Java

Развитие языков программирования

- 1960-е Fortran, Algol, Basic, Cobol, Lisp
- 1970-е Pascal, C, Simula, Smalltalk, Prolog
- 1980-е C++, Object Pascal, Ada, Occam
- 2000-е C#

Уровень абстракции языка

- Ассемблер
- Fortran
- C
- Pascal
- C++
- C#



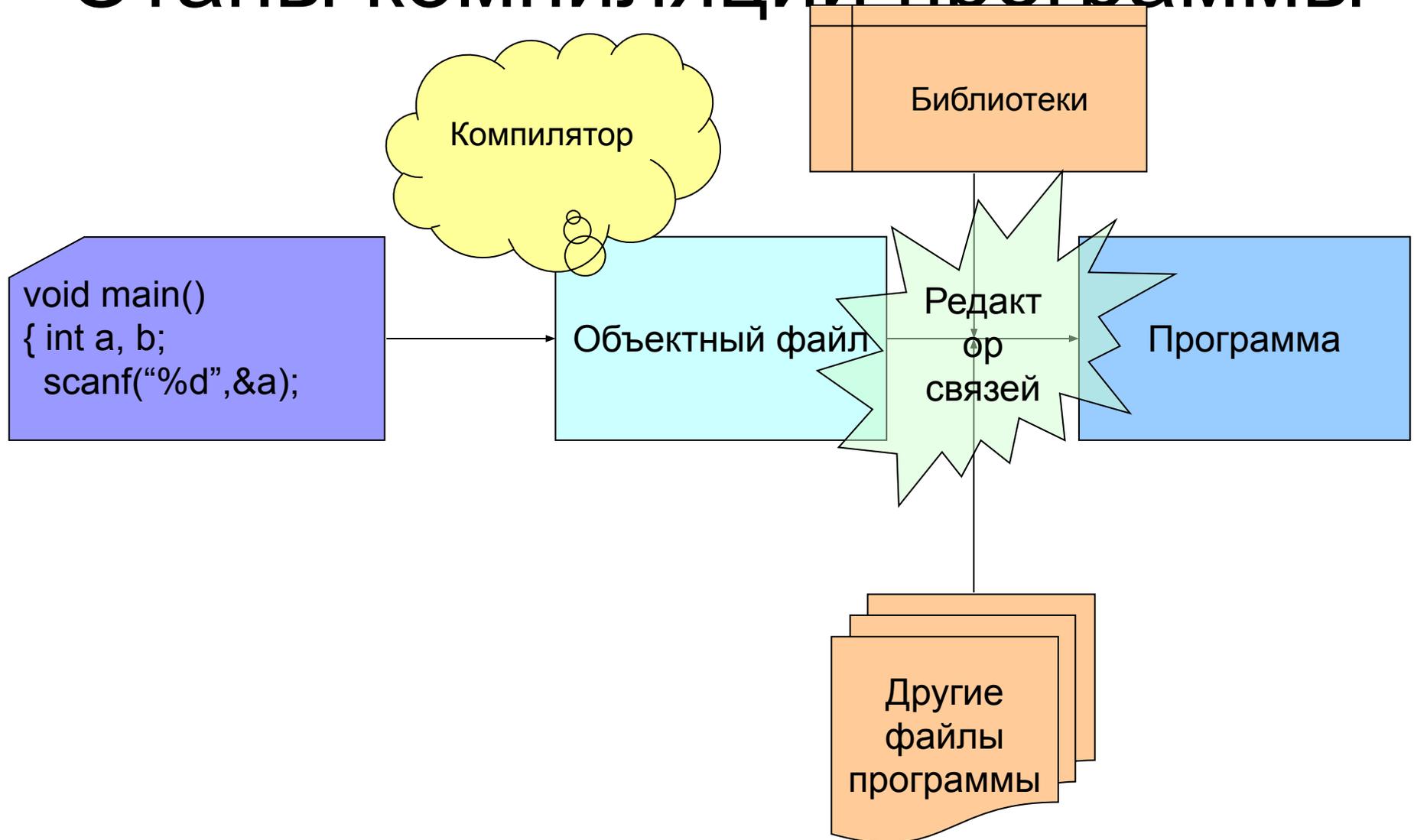
Машинный код и язык ассемблера

| | | | |
|----|----|----|---------|
| 00 | 61 | | LD 1 |
| 01 | 62 | | LD 2 |
| 02 | 10 | | ADD |
| 03 | 53 | 06 | GT A |
| 05 | 18 | | SGN |
| 06 | 41 | | A: ST 1 |
| 07 | 50 | | STOP |

Трансляторы

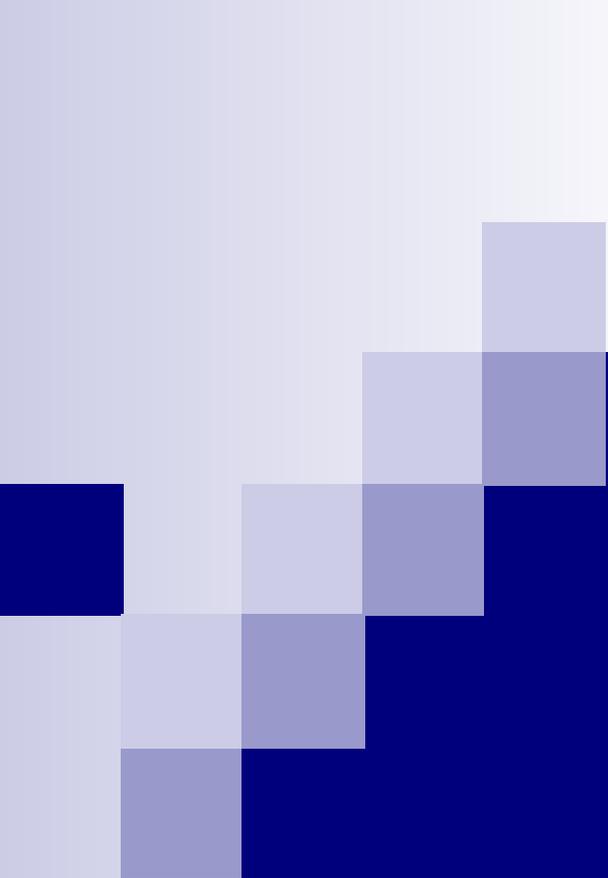
- Компилятор + компоновщик
(C/C++, Pascal)
- Интерпретатор
(Basic, JavaScript, HTML)
- Использование промежуточного языка и виртуальной машины
(C#, Java)

Этапы компиляции программы



Этапы разработки программ

- Постановка задачи
- Определение структур данных
- Проектирование алгоритма
- Кодирование
- Отладка
- Тестирование
- Доводка



Язык С

Лекция №1

Переменные, типы данных,
операция присваивания

Пример программы

```
#include <stdio.h>
```

```
/* Пример №1 – простейшая программа */
```

```
void main()
```

```
{
```

```
    int year, month;
```

```
    year = 2011;    month=10;
```

```
    printf("Сейчас %d год", year);
```

```
}
```

Идентификаторы и переменные

- **Идентификатор** (имя) – последовательность букв и цифр, начинающаяся с буквы (регистр в C/C++ различается!)
- **Переменная** – величина, изменяющая своё значение во время выполнения программы, обозначается идентификатором, всегда принадлежит какому-либо типу данных
- **Тип данных** – множество значений, которое может принимать переменная (а также множество операций, применимых к ней)

Целочисленные типы данных

| Тип | Размер | Диапазон |
|---------------|--------|----------------------------------|
| unsigned char | 1 | 0 ... 255 |
| char | 1 | -128 ... +127 |
| short int | 2 | -32 768 ... +32 767 |
| unsigned int | 4 | 0 ... 4 294 967 295 |
| int | 4 | -2 147 483 648 ... 2 147 483 647 |
| unsigned long | 4 | 0 ... 4 294 967 295 |
| long | 4 | -2 147 483 648 ... 2 147 483 647 |
| long long | 8 | $\pm 2^{63}-1$ |

Вещественные типы данных

| Тип | Размер | Точность | Диапазон |
|-------------|--------|----------|-------------------------------|
| float | 4 | 6-7 | $\pm 3.4 \cdot 10^{\pm 38}$ |
| double | 8 | 15-16 | $\pm 1.7 \cdot 10^{\pm 308}$ |
| long double | 10 | 19-20 | $\pm 3.4 \cdot 10^{\pm 4932}$ |

| | |
|------------------------------|----------------|
| Мантисса 1234567890123456 | Порядок +15 |
|------------------------------|----------------|

0.1234567890123456 · 10⁺¹⁵

Описание переменных

- `int i, j;`
- `float x;`
- `double a, b, c;`

Запись непосредственных констант

| Тип константы | Пример |
|---------------|--|
| int | 12, +45, -1567439 |
| int | 055 (8-я система), 0x1A (16-я система) |
| unsigned int | 600000, 123432135 |
| long int | 5L, 100000 |
| short int | 10, 32H, -10h |
| float | 2.12, 1.0, 1234.0f, 3E+6, 2.1E-4f |
| double | 123.43, 6.67E-34 |
| long double | 1.39L, 12.0l |
| char | 'a', '\n', '1', '\\" |

Записи специальных констант

`\n` Новая строка

`\r` Возврат каретки

`\t` Горизонтальная табуляция

`\"` Кавычка

`\'` Апостроф

`\\` Обратная косая черта

`\0` Нулевой символ

`\123` Символ с восьмеричным кодом 123

`\xA1` Символ с шестнадцатеричным кодом A1

Операция присваивания

- `n=1;`
- `n=k*5;`
- `n=n+1;`
- `a=2.99792E+10;`
- `x=a/1000.0`
- `i=j=0;` `i=(j=0);`

Соответствие типов данных

- Вообще говоря, тип переменной слева от знака присваивания должен соответствовать типу выражения!
- Но есть несколько исключений
- Возможно явное приведение типов

Автоматическое приведение типов

```
int i;  
float x;  
double y;  
char c;
```

```
x = i; // Допустимо
```

```
i = x; // Предупреждение или ошибка
```

```
c = i; // Допустимо, но возможна потеря битов
```

```
x = y; // Допустимо, но будет потеря точности
```

Явное приведение типов

```
int i;  
float x;  
double y;  
char c;
```

```
x = (float) i;
```

```
i = (int) x;
```

```
i = int(y);
```

Неверное понимание приведения типов –
источник трудно выявляемых ошибок в программе

Инициализация переменных

- `int i = 0, j = 0;`
- `float x = 1.0;`
- `double a, b, c = 2.718281828;`

Использование неинициализированных переменных – источник трудно выявляемых ошибок в программе

Бинарные операции

+ – сложение

– – вычитание

* – умножение

/ – деление

% – деление по модулю

Целочисленные арифметические выражения

- n
- $25*n+3$
- $25*(n+3)$
- $n/2$ – деление на цело! $5 / 2 = 2$
- $n\%2$ – остаток от деления $5\%2 = 1$

Унарные операции

- – унарный минус
- ++ – увеличение на единицу
- – уменьшение на единицу

```
i=1;  
j=i++; // j=1  
k=++i; // k=3
```

Вещественные арифметические выражения

- x
- $25.0 * x + 3$
- $25 * (x + 3.0)$
- $x / 2$ – вещественное деление $5.0 / 2 = 2.5$
- $2 * \sin(x)$

Операции присваивания

- = Прямое присваивание значения левому операнду
- += Складывает значения левого и правого операндов и присваивает результат левому операнду
- = Вычитает значения левого и правого операндов и присваивает результат левому операнду
- *= Умножает значения левого и правого операндов и присваивает результат левому операнду
- /= Делит значения левого и правого операндов и присваивает результат левому операнду

Примеры операций присваивания

```
x=a+(b=c-d++);
```

```
m*=2;
```

```
max = (x>y) ? x : y;
```

```
a=sin(x)*cos(y);
```

```
c=sqrt(2.0);
```

Приоритеты операций

Операторы

1. () [] -> :: .
2. ! ~ + - ++ -- & * (typecast) sizeof new delete
3. .* ->*
4. * / %
5. + -
6. << >>
7. < <= > >=
8. == !=
9. &
10. ^
11. |
12. &&
13. ||
14. ?: (условное выражение)
15. = *= /= %= += -= &= ^= |= <<= >>=
16. ,

Ассоциативность

- Left to right
- Right to left
- Left to right
- Right to left
- Right to left
- Left to right

Простейшая программа

```
#include <stdio.h>
```

```
void main()
```

```
{ float R;
```

```
float L, S;
```

```
printf("Enter R ");
```

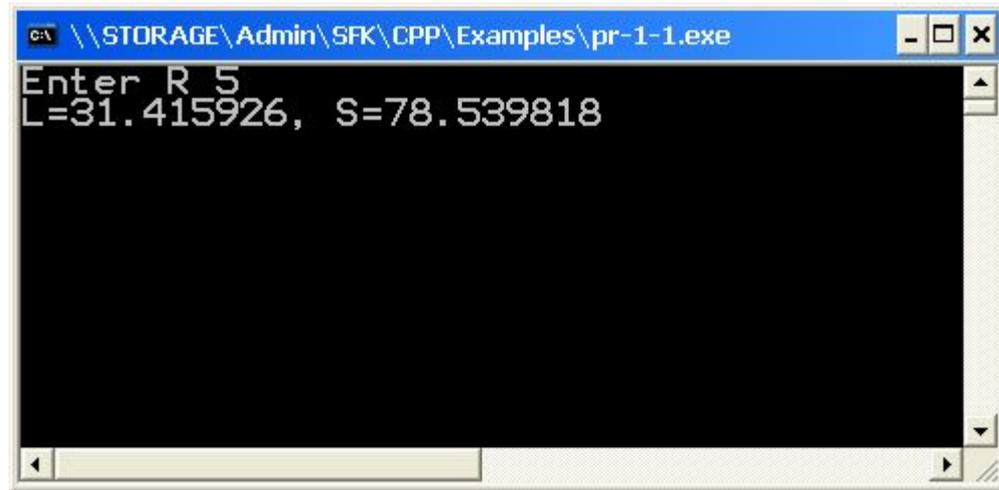
```
scanf("%f", &R);
```

```
L=3.1415926*2*R;
```

```
S=3.1415926*R*R;
```

```
printf("L=%f, S=%f", L, S);
```

```
}
```



```
C:\ \\STORAGE\Admin\SFK\CPP\Examples\pr-1-1.exe
Enter R 5
L=31.415926, S=78.539818
```

Функция printf

```
printf("Hello, World!\n");  
printf("x=%f, y=%f\n", x, y);  
printf("%05d", 15); // Выдаст 00015
```

%c – символ

%d – целое десятичное число

%i – целое десятичное число

%o – целое восьмеричное число

%x – целое шестнадцатеричное число (a1)

%X – целое шестнадцатеричное число (A1)

%u – беззнаковое десятичное

%f – вещественное число xx.xxx

%F – вещественное число xx.xxx

%e – вещественное число x.xx e+ xx

%E – вещественное число x.xx E+ xx

%G – %F или %E (что компактнее)

%g – %f или %e (что компактнее)

%s – строка символов

%p – указатель

%% – символ %

Модификаторы l и h (L, H) %ld – long int, %hu – short unsigned, %Lf – long double

Модификаторы точности %8.2F, %16E, %8d

Функция scanf

`%d` – целое десятичное число

`%i` – целое десятичное число

`%o` – целое восьмеричное число

`%x` – целое шестнадцатеричное число

`%h` – чтение short int

`%e` – чтение числа типа float

`%c` – СИМВОЛ

`%s` – чтение строки СИМВОЛОВ

`%p` – указатель

```
scanf("%d%*c%d", &i, &j);
```

20+50

$i \rightarrow 20, j \rightarrow 50, *$ – прочитайте данные указанного типа, но проигнорируйте их

```
scanf("%5s", &s); // Максимальная длина поля
```

```
scanf("%dplus%d", &i, &j); // Проигнорировать текст plus
```