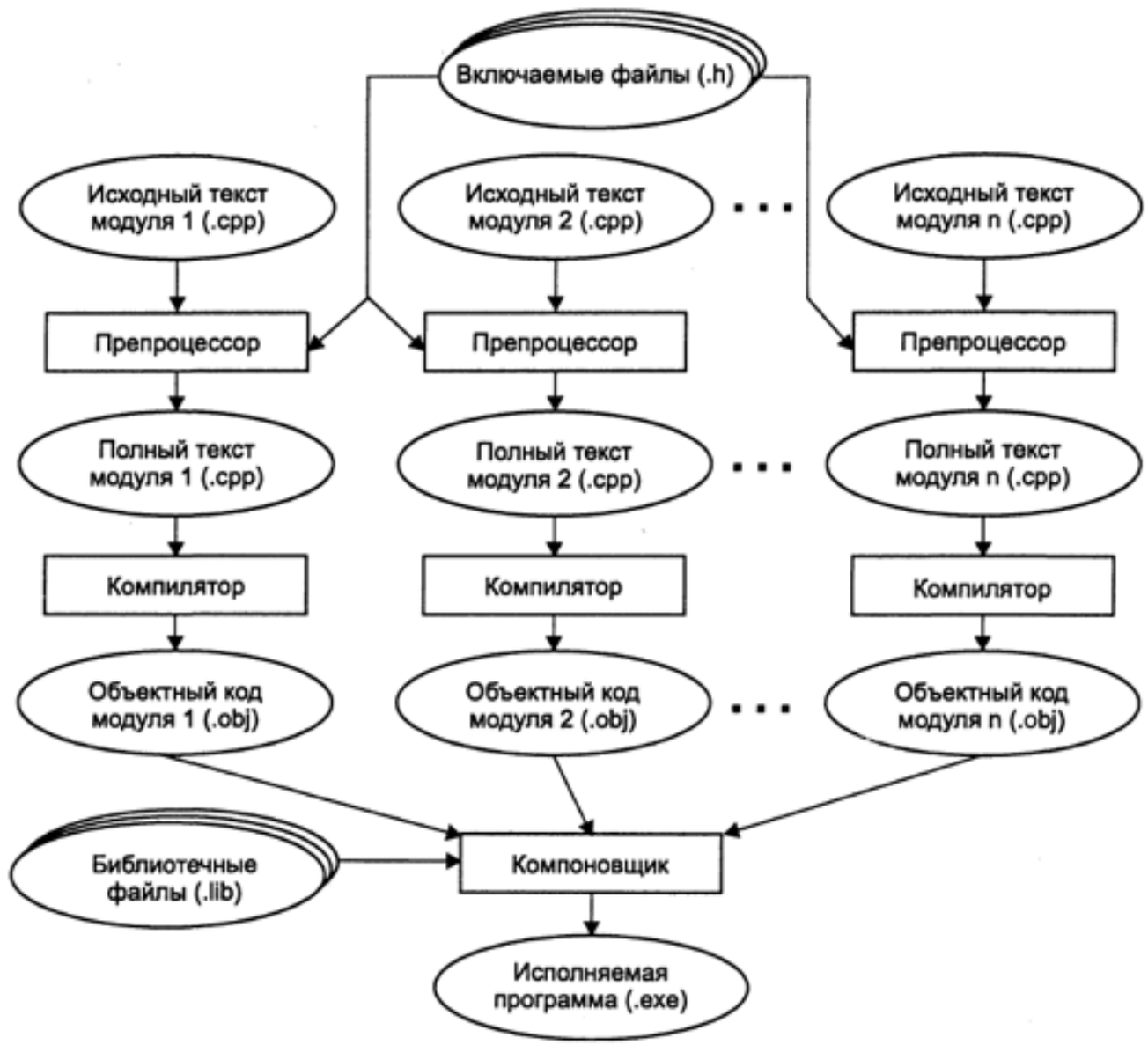
Алгоритмы и структуры данных

Семинар 3

Приведение типов

- static_cast
- dynamic_cast
- const_cast (чаще всего зло!)
- reinterpret_cast

Процесс компиляции программы



Этапы создания исполняемой программы

Парадигмы программирования

- Обобщенное программирование: решите, какие нужны алгоритмы; параметризируйте их так, чтобы они могли работать со множеством подходящих типов и структур данных.

Обобщенное программирование

```
int max(int a, int b)
{
  return a > b ? a : b;
}

float max(float a, float b)
{
  return a > b ? a : b;
}
```

Шаблон функции

```
template<typename T>
const T& max(const T& a, const T& b)
{
  return a > b ? a : b;
}

max<int>(1, 2);
max<float>(1.0, 2.0);
```

Шаблон функции

```
template<class T>
const T& max(const T& a, const T& b)
{
  return a > b ? a : b;
}

max(1, 2);
max(1.0, 2.0);
max(1, 2.0); // Не скомпилируется
```

Параметры шаблона

- Типы

- Значения перечислимых типов: перечисления и целые числа.

Параметры шаблона

```
template<int val>
int add(int num)
{
  return num + val;
}

add<2>(3); // == 5;
```

Параметры шаблонов

```
template<class T, class U = T>
U delta(const T& a, const T& b)
{
  return a - b;
}

Date date1, date2;
delta<Date, TimeDelta>(date1, date2);
```

Специализация шаблона

```
class BigInt
{
public:
   ...
   bool compare(const BigInt& a, const BigInt& b)
   {
      return ...
   }
   ...
private:
   ...
};

BigInt a, b;
max(a, b);
```

Специализация шаблона

```
template<>
const BigInt& max<BigInt>(const BigInt& a, const BigInt& b)
{
   return a.compare(b) ? b : a;
}

BigInt a, b;
max(a, b);
```

std::sort

```
template<class Iter, class Cmp>
void sort(Iter a, Iter b, Cmp cmp);

template<class T>
class Less
{
public:
   void operator()(const T& a, const T& b)
   {
      return a < b;
   }
};

int a[100];
std::sort(a, a + 100, Less<int>());
```

Шаблон класса

```
template<class T>
class Array
{
public:
   T& operator[]()
   {
      return ...
   }
   const T& operator[]() const
   {
      return ...
   }
private:
}
```

Частичная специализация шаблона

```
template<class T, class Allocator = DefaultAllocator>
class Vector
{
public:
   static const bool value = false;
};


template<class Allocator>
class Vector<bool, Allocator>
{
public:
   static const bool value = true;
};
```

Black magic

- Давайте сделаем что-то нестандартное?
  Посчитаем факториал на этапе компиляции :)