ЛЕКЦИЯ 12
ПЕРЕГРУЗКА ФУНКЦИЙ

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

ЛЕКТОР ФУРМАВНИН С.А.

ПОНЯТИЕ ПЕРЕГРУЗКИ

В широком смысле **перегрузка** (overloading) — это возможность одновременно использовать несколько функций с одним именем. Компилятор различает их благодаря тому, что они имеют разный набор параметров. В точки вызова компилятор анализирует типы аргументов и определяет, какая конкретно функция должна быть вызвана. В русскоязычной литературе иногда можно встретить термин «совместное использование», но, похоже, он не прижился.

ПОНЯТИЕ ПЕРЕГРУЗКИ

Функции (а также шаблоны функций) называются **перегруженными** (overloaded), если они объявлены в одной области видимости (scope) и имеют одно и то же имя.

РАЗРЕШЕНИЕ ПЕРЕГРУЗКИ

Правила выбора наиболее подходящей функции (overload resolution rules) при попытке полного и формального описания могут оказаться весьма сложными и запутанными (это из тех вещей, которые до конца знают только разработчики компилятора), но как это часто бывает, во многих практически значимых случаях они являются интуитивно понятными и особых проблем у программиста не вызывают.

РАЗРЕШЕНИЕ ПЕРЕГРУЗКИ

При разрешении перегрузки компилятор прежде всего должен выбрать область видимости, в которой и будет выполнятся разрешение перегрузки. Такая область видимости называется текущей. Если в текущей области видимости нет ни одной функции с искомым именем, текущей областью видимости становится объемлющая область видимости. Но, если в текущей области видимости найдена хотя бы одна функция с искомым именем, то выполняется разрешение перегрузки в данной области видимости и объемлющая область видимости рассматриваться не будет.

РАЗРЕШЕНИЕ ПЕРЕГРУЗКИ

Компилятор затем ищет функции-кандидаты. Затем из кандидатов выбирается самый лучший. Выигрывает тот кандидат, для подстановки параметров в который требуется наименьшее количество преобразований.

КРИТЕРИИ РАЗРЕШЕНИЯ ПЕРЕГРУЗКИ

Критерии применяются по очереди:

1. Точное совпадение. Т.е. совпадение без преобразований или только с тривиальными преобразованиями (`T` в `const T`, имя массива в указатель на первый элемент)
2. Совпадение на продвижениях. Т.е. с интегральными продвижениями
3. Совпадение на стандартных преобразованиях.
4. Совпадение на пользовательских преобразованиях
5. Совпадение с использованием ...

УПРАЖНЕНИЕ.

Напишите функцию, которая принимает два числа: int и int.

Функция должна возвращать результат следующего выражения:

$$f(x) = x + 2x + 3x + \cdots + (n-1)x + nx$$

Перегрузите эту функцию для double таким образом, чтобы она возвращала значение следующего выражения:

$$f(x) = \lfloor x \rfloor + 2\lceil x \rceil + 3\lfloor x \rfloor + \cdots$$

n раз

ОБОБЩЕНИЕ ФУНКЦИЙ

Пожалуй, единственным способом написать на C максимум двух чисел является макрос

```
#define MAX(x, y) (((x) > (y)) ? (x) : (y));
```

Перечислите все проблемы в этом макросе
На C++ шаблон функции лишен этих проблем

```
template <typename T> T max(T x, T y) {
  return (x > y)? x : y;
}
```

ОБОБЩЕНИЕ ВМЕСТО VOID*

Стандартная функция из библиотеки C

```
void qsort (void* base, size_t num, size_t size, int
(*compr)(const void*, const void*));
```

Что можно с ней сделать, используя шаблоны?

ОБОБЩЕНИЕ ВМЕСТО VOID*

Первая итерация

```
template <typename T, typename Comp>
void qsort (T* base, size_t num, Comp compare);
```

Вместо передачи указателя и длины, можно передавать два указателя на начало и конец интервала

ОБОБЩЕНИЕ ВМЕСТО VOID*

Вторая итерация

```
template <typename T, typename Comp>
void qsort (T* start, T* fin, Comp compare);
```

Вместо передачи указателей можно использовать указателе-подобные объекты, так называемые итераторы и получить

```
template <typename It, typename T, typename Comp = std::less>
void sort (It start, It fin, Comp compare);
```

ВОЗВОДИМ ЧИСЛО В СТЕПЕНЬ

“The first step is to get the algorithm right. The second step is to figure out which sorts of things (types) it works for” – Alex Stepanov

Начнем с первого

```
unsigned nth_power(unsigned x, unsigned n); // returns x^n
```

Как написать тело этой функции?

ВЫБИРАЕМ ПРАВИЛЬНЫЙ АЛГОРИТМ

“The first step is to **get the algorithm right**. The second step is to figure out which sorts of things (types) it works for” – Alex Stepanov

```
unsigned nth_power(unsigned x, unsigned n) {
  unsigned acc = 1;
  if ((x < 2) || (n == 1)) return x;
  while (n > 0) {
    if ((n & 0x1) == 0x1) { acc *= x; n -= 1;}
    else { x *= x; n /= 2; }
  return acc;
}
```

Разумеется, мы вариант перемножить x ровно n раз в цикле даже не рассматриваем

ИЩЕМ ВОЗМОЖНОСТИ ОБОБЩЕНИЯ

“The first step is to get the algorithm right. The second step is to figure out which sorts of things (types) it works for” – Alex Stepanov

```
unsigned nth_power(unsigned x, unsigned n) {
  unsigned acc = 1;
  if ((x < 2) || (n == 1)) return x;
  while (n > 0) {
    if ((n & 0x1) == 0x1) { acc *= x; n -= 1;}
    else { x *= x; n /= 2; }
  return acc;
}
```

Как обобщить алгоритм?

НАИВНОЕ ОБОБЩЕНИЕ

“The first step is to get the algorithm right. The second step is to figure out which sorts of things (types) it works for” – Alex Stepanov

```
template <typename T> T nth_power(T x, unsigned n) {
  T acc = 1;
  if ((x < 2) || (n == 1)) return x;
  while (n > 0) {
    if ((n & 0x1) == 0x1) { acc *= x; n -= 1;}
    else { x *= x; n /= 2; }
  return acc;
}
```

Как обобщить алгоритм?

КЛАССЫ И СТРУКТУРЫ

Структура в C – это набор похожих или различающихся типов данных.

```
struct Point {
      int x, y;
}
```

C++ расширяет возможности структур, позволяя включать в структуры функции. Объединение данных и функций, их обрабатывающих, в единое целое является краеугольным камнем **объектно-ориентированного программирования**.

КЛАССЫ И СТРУКТУРЫ

В C++ к структурам добавляется еще один элемент – класс, который также может хранить данные и функции.

В синтаксисе структуры и класса почти нет различий, и потому они почти всегда взаимозаменяемы.

Однако многие программисты используют структуры для хранения данных, а классы – для хранения данных и функций.

КЛАССЫ И СТРУКТУРЫ

```
struct Rectangle {
      double width, height;
      double area() { return width * height; }
 };

class Rectangle {
      double width, height;
      double area() { return width * height; }
 };
```

КЛАССЫ И СТРУКТУРЫ

Слова `class` и `struct` — ключевые, за ним следует слово Rectangle, являющееся именем класса (структуры). Аналогично функциям и пространствам имен структурное тело класса (функции) помещается в фигурные скобки, которые ограничиваются точкой с запятой.

КЛАССЫ И СТРУКТУРЫ

Переменные, объявленные в классе, называются **членами данных** или **полями класса**.

Функции, объявленные в классе, называются **членами-функциями** или **методами класса**.

КЛАССЫ И СТРУКТУРЫ. РАЗМЕР

```
struct S {
	int n;
};

int main() {
	S s;
	std::cout << sizeof(s) << std::endl;
	return 0;
}
```

Что на экране?

КЛАССЫ И СТРУКТУРЫ. РАЗМЕР

```
struct S {
	int x, y;
};

int main() {
	S s;
	std::cout << sizeof(s) << std::endl;
	return 0;
}
```

Что на экране?

КЛАССЫ И СТРУКТУРЫ. РАЗМЕР

```
struct S {
	int x;
	char c;
};

int main() {
	S s;
	std::cout << sizeof(s) << std::endl;
	return 0;
}
```

Что на экране?

КЛАССЫ И СТРУКТУРЫ. РАЗМЕР

```
struct S {
	int x;
	double d;
	char c;
};

int main() {
	S s;
	std::cout << sizeof(s) << std::endl;
	return 0;
}
```

Что на экране?

КЛАССЫ И СТРУКТУРЫ. РАЗМЕР

```
struct S {
	int x;
	char c;
	double d;
};

int main() {
	S s;
	std::cout << sizeof(s) << std::endl;
	return 0;
}
```

Что на экране?

КЛАССЫ И СТРУКТУРЫ

В общем в структурах выравниваются в памяти поля по границе кратной своему же размеру. То есть 1-байтовые поля не выравниваются, 2-байтовые — выравниваются на чётные позиции, 4-байтовые — на позиции кратные четырём и т.д.

КЛАССЫ И СТРУКТУРЫ

```
struct Rectangle {
private:
      double width, height;
public:
      double area() { return width * height; }
 };

class Rectangle {
private:
      double width, height;
public:
      double area() { return width * height; }
 };
```

КЛАССЫ И СТРУКТУРЫ

Изучим два незнакомых ранее вам ключевых слова: `private` и `public`. Они позволяют управлять доступам к членам класса.

Закрытые члены (**private members**) недоступны за пределами класса, а открытые члены (**public members**) – доступны.

Обычно члены данных (поля класса) размещают в закрытой (private) области, а функции-члены – в открытой (public) области класса.

КЛАССЫ И СТРУКТУРЫ

Таким образом члены данных находятся в безопасности и не могут быть изменены произвольно пользователем, а класс сохраняет свой инвариант. А функции-члены, доступны пользователю класса, обеспечивают корректный и ограниченный доступ к членам данных.

Концепция обеспечения безопасности членов данных и предоставления им доступа через функции-члены называется **сокрытием данных** или **инкапсуляцией**.

Поскольку функции-члены обеспечивают систематический доступ к скрытым членам данных, их набор называют **интерфейсом класса**.

КЛАССЫ И СТРУКТУРЫ

Объект класса или структуры называют **экземпляром класса** (**instance**), а операцию создания объекта класса – **инстанцированием** (instatiation)

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Дорохова Т.Ю., Основы алгоритмизации и программирования : учебное пособие для СПО / Т.Ю. Дорохова, И.Е. Ильина. – Саратов, Москва : Профобразование, Ай, Пи Ар Медиа, 2022. – 139 с.
2. Кудинов Ю.И., Основы алгоритмизации и программирования : учебное пособие для СПО / Ю.И. Кудинов, А.Ю. Келина. – 2-е изд. – Липецк, Саратов: Липецкий государственный технический университет, Профообразование, 2020. – 71 с.
3. Дональд Кнут, Искусство программирования. Том 1. Основные алгоритмы / Ю.В. Козаченко. - 3-е изд – Москва, Санкт-Петербург: ВИЛЬЯМС, 2018. – 721 с.