Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра информационных систем и технологий

**Математическое программирование**

**Курс лекций**

**для студентов специальности:**

1-40 01 02-03 – «Информационные системы и технологии (издательско-полиграфический комплекс)»

Составитель:

Старший преподаватель,

кандидат технических наук

А. И. Бракович

Минск 2010

**Лекция 1**

1,2:

**КУРС «МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ»**

|  |  |
| --- | --- |
| Лекции | **34 часа (**17 лекций**)** |
| Лабораторные работы | **18 часов (**9 лабораторных работ**)** |
| Самостоятельная работа | **78 часов (**выполнение лабораторных**)** |
| **Всего** | **130 часов** |
| Экзамен | **2 семестр** |

3:

**ВВЕДЕНИЕ**

**Математическое программирование** –область математики, разрабатывающая теорию и численные методы решения многомерных экстремальных задач, т.е. задач на экстремум функции многих переменных с ограничением на область определения.

Решение задачи математического программирования осуществляется в 4 этапа.

1. Построение математической модели.
2. Классификация задачи.
3. Выбор метода решения.
4. Вычисление.

4:

В общем виде **модель задачи математического программирования** выглядит следующим образом:

|  |  |
| --- | --- |
| , , где | |
|  | – искомая, в общем случае векторная, величина; |
|  | – область определения искомой величины; |
|  | – функция цели (функция определяющая значение критерия оптимальности); |

В зависимости от природы множества  и вида функции  задачи математического программирования **классифицируются** как задачи

* дискретного программирования (комбинаторная оптимизация) ( конечно или счетно);
* целочисленного программирования ( подмножество множества целых чисел);
* линейного программирования (– линейная функция,  – может быть определено с помощью линейных неравенств);
* нелинейного программирования (– нелинейная функция и/или в описании  присутствует хотя бы одна нелинейная функция);
* векторная оптимизация (– векторная функция).

Кроме того, разделами математического программирования являются динамическое, стохастическое и параметрическое программирование, сетевое планирование, потоки в сетях и т.д.

**Метод решения** задачи математического программирования определятся в зависимости от исходных данных.

**Вычисление** решения задачи математического программирования осуществляется, как правило, с помощью компьютерной техники.

5:

**СМЕЖНЫЕ ДИСЦИПЛИНЫ**

Модели

Модели

Методы оптимизации

6:

**РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА**

|  |  |
| --- | --- |
| 1. | Смелов, В. В. Комбинаторные алгоритмы оптимизации : учеб.-метод. пособие для студентов специальности «Информационные системы и технологии» (издательско-полиграфический комплекс) / В. В. Смелов, А. И. Бракович – Минск : БГТУ, 2010. |
| 2. | Смелов, В. В. Основы сетевого планирования: учеб.-метод. пособие для студентов специальности «Информационные системы и технологии (издательско-полиграфический комплекс»/ В. В. Смелов, Т. П. Брусенцова. – Минск: БГТУ, 2010. – 231 с. |
| 3. | Костевич Л.С. Математическое программирование. – Мн.: Новое знание, 2003, – 424 с. |
| 4 | Таха Х.А. Введение в исследование операций. – М.: Вильямс, 2001. – 912 с. |
| 5. | Кузнецов А.В., Сакович В.А., Холод Н.И. Высшая математика. Математическое программирование. –Мн.: Высш.шк., 1994. – 288 с. |
|  |  |

**ПРИМЕРЫ ЗАДАЧ МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ**

**Задача о кратчайшем расстоянии между вершинами графа**

**Задача о рюкзаке**

**Задача о коммивояжере (о бродячем торговце)**

**Задача о нахождении максимального потока в сети**

**Задача линейного программирования**

**Транспортная задача**

**Задача нелинейного программирования**

**Векторная оптимизация**

**Сетевое планирование**

7:

**Вспомогательные функции**

Алгоритмы, в основном мы будем оформлять в виде функции. Целью первой лабораторной работы является приобретение навыков составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления.

**Генерация случайных чисел**

// # include <cstdlib>

**void srand (**

**unsigned int s** // [in] стартовое число генератора

**)**

Директивы препроцессора представляют собой инструкции, записанные в исходном тексте программы и предназначенные для выполнения препроцессором языка. Фактически это часть компилятора, которая умеет исполнять директивы. #include вставляет текст файла, указанного далее.

Назначение функции srand – установка начального значения псевдослучайного числа.

// # include <cstdlib>

**int rand ()**

Функция rand возвращает псевдослучайное целоче число от 0 до RAND\_MAX. RAND\_MAX это положительная константа, определенная с помощью директивы # include <cstdlib>.

8:

**Функции времени**

// # include <ctime>

**time\_t time (**

**time\_t\* t** //[out] указатель на буфер (8 байт)

**)**

Функция time\_t возвращает количество секунд прошедшие с 00:00:00 01.01.1970 к точке вызова или/и в буфер, если параметр t не NULL.

// # include <ctime>

**clock\_t clock()**

//--(1 сек = **CLOCKS\_PER\_SEC** единиц)

Функция clock\_t возвращает количество единиц процессорного времени или тактовую частоту (сек = CLOCKS\_PER\_SEC единиц) прошедших с момента старта приложения. Константа CLOCKS\_PER\_SEC показывает сколько единиц процессорного времени находится в одной секунде.