Карта мира с произвольным центральным меридианом в MapInfo

Обсудить в форуме Комментариев — 4

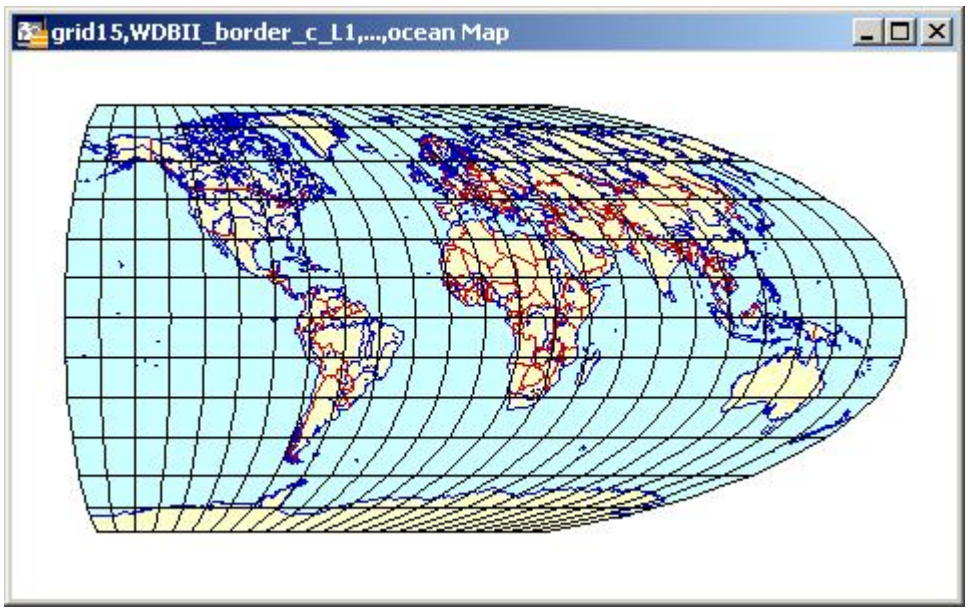
Эта страница опубликована в основном списке статей сайта
по адресу http://gis-lab.info/qa/mapinfo-worldmap-custom-meridian.html

Статья представляет собой пошаговое руководство для пользователей MapInfo по подготовке карты мира для отображения в проекциях с меридианом, отличающимся от среднего меридиана имеющихся слоёв.

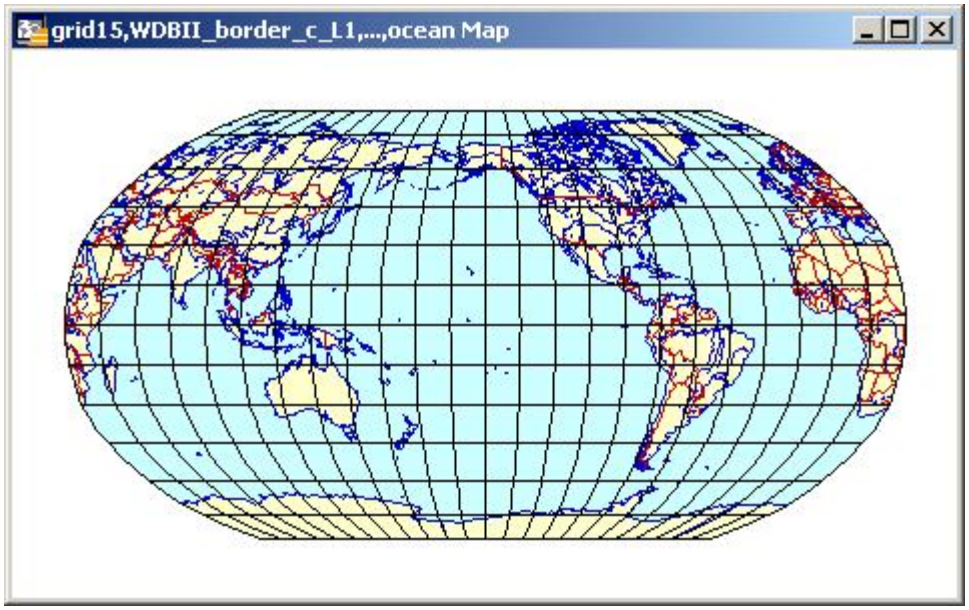
Содержание

Введение

Опишем проблему парой картинок. Предположим, нужно отобразить стандартную карту мира в проекции Робинсона с центральным меридианом 150° з. д.



Так рисует карту MapInfo



Такую карту хотелось бы видеть

Очевидно, MapInfo, как многие другие ГИС, при отображении карты в какой-либо проекции не оборачивает отображаемые слои вокруг меридиана-антипода, который отстоит на 180° от центрального. Это позволяет создавать карты мира, разрезанные по произвольной линии, например, линии смены дат. Однако для пользователя свобода, как обычно, идёт рука об руку с дополнительными хлопотами. Если имеющаяся в наличии карта должна быть отображена в форме милой глазу симметричной фигуры в проекции, центральный меридиан которой отличается от среднего меридиана исходной карты, то картографу для этого придётся приложить некоторые усилия.

Данные

В качестве тестового материала используем карту мира GSHHG, которая распространяется под лицензией LGPL. Эта карта развивается как географическая основа открытого проекта GMT. GMT умеет оборачивать данные вокруг меридиана-антипода, поэтому слои карты в «родном» формате не содержат разрезанных объектов. Однако при экспорте слоёв в формат ESRI shapefiles полигоны под меридианом 180° разрезаются на восточную и западную часть, что даёт карту в стандартном диапазоне долгот ±180°. Для демонстрации возьмём из GSHHG несколько слоёв грубого (crude) разрешения.

Постановка задачи

Наша задача - отобразить карту в паре проекций с центральным меридианом 150° з. д. Для достижения этой цели создадим новые слои в диапазоне долгот от 330° з. д. до 30° в. д. Кроме того, дополним карту слоями сетки параллелей и меридианов **grid15** и «океана» **ocean**.

Построение карты с помощью программы PacWorld

Утилита PacWorld от IAA Pty Ltd решает задачу преобразования карты из стандартного диапазона долгот в диапазон 0°–360°. Поскольку она доступна в кодах MapBasic, можно модифицировать её для работы с произвольным центральным меридианом.

Алгоритм работы начинается с создания полигона, одной из сторон которого является начальный меридиан. Объекты слоя разрезается этим полигоном. Затем объекты западного полушария модифицируются: каждый узел перемещается на 360° к востоку.

Недостатки **PacWorld**, помимо упомянутой жёсткой привязки к меридиану 180° в. д.:

- разрезы материков на краю карты отображаются в проекциях несглаженными прямыми отрезками; — легко ис, запрограммировав вставку промежуточных узлов в сторону полигона, образованную начальным меридианом;
- некорректно трансформируется Антарктида; — можно обойти эту проблему через конвертирование

полигона в полилинию и обратно с некоторым редактированием;
- зависание при переносе объектов с большим количеством узлов; с картой GSHHG это происходит уже для слоя континентов со средним (intermediate) разрешением; похоже, дело в принципиальной ограниченности ресурсов, выделяемых программам MapBasic; — фатальный недостаток.

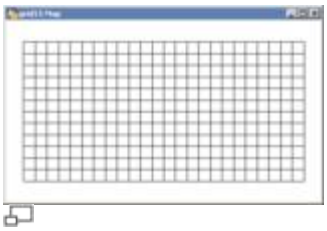
Главная проблема с зависанием связана не с процедурой разрезания. Зависание происходит в процессе перемещения узлов. Можно модифицировать **PacWorld** так, чтобы он только разрезал объекты, а перемещение осуществить другими средствами. Однако с методической точки зрения полезно поработать руками.

Построение карты вручную

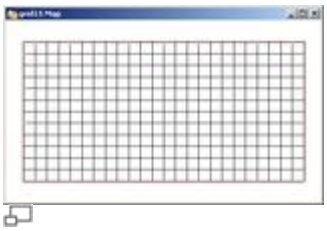
Конструирование координатной сетки и «океана»

Построим слой параллелей и меридианов **grid15** программой **GridMaker**, входящей в набор стандартных утилит MapInfo. Используем четыре линии по контуру для создания слоя «океана» **ocean**.

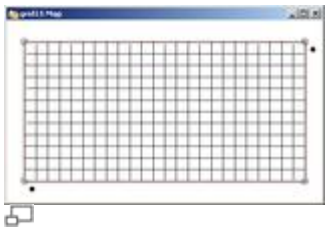
Диалог GridMaker: параметры введены



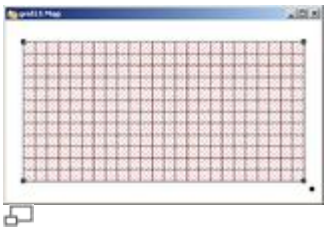
Слой **grid15** создан



Выделим крайние линии



Скопируем в косметический слой

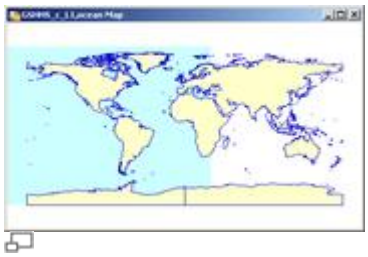


Объединим и превратим в полигон

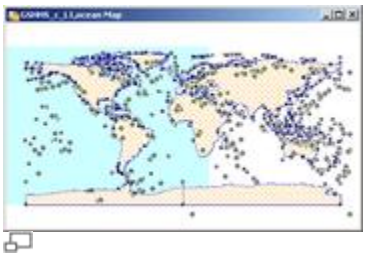
Сохраним косметику в слой **ocean**

Разрезание слоя

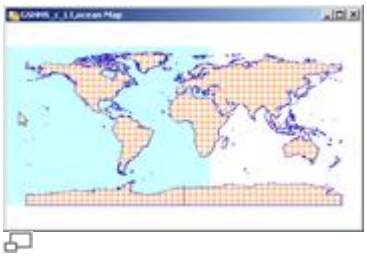
Разрежем слой континентов и островов **GSHHS_c_L1** на две половины полигоном слоя **ocean**. Часть, попадающую на полигон (и, следовательно, на будущую карту), сохраним как новый слой с прежним именем **GSHHS_c_L1** в другую папку. Часть, не попадающую на полигон, экспортируем в файл формата MIF/MID под именем **GSHHS_c_L1_1.MIF**.



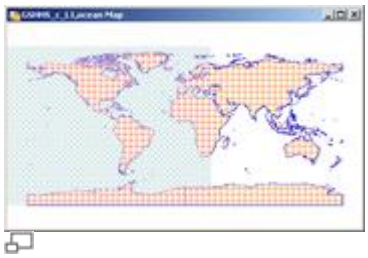
Откроем **ocean** и исходный слой островов **GSHHS_c_L1**



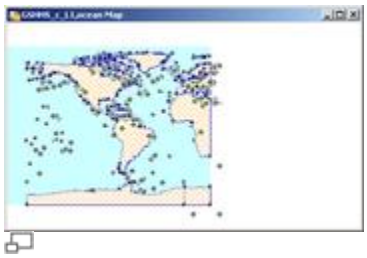
Выделим все объекты слоя островов



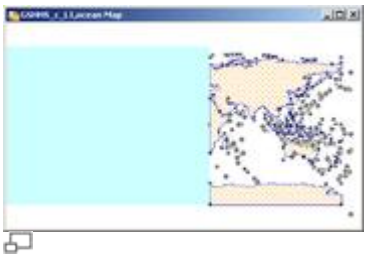
Перейдём в режим выбора изменяющих объектов



Укажем **ocean** в качестве изменяющего объекта



После команды Erase Outside сохраним объекты в новый слой **GSHHS_c_L1**; восстановим таблицу



Повторим действия с выделениями; после команды Erase экспортируем объекты в файл MIF

Перемещение половины слоя на 360°

Чтобы объекты из файла **GSHHS_c_L1_1.MIF** оказались на своих местах на будущей карте, их необходимо перенести на 360° к западу. К сожалению, перенести объекты ровно на заданное число градусов по долготе или широте в MapInfo практически невозможно. Мы слукавили, говоря о построении карты вручную от начала и до конца.

Приведём листинг программы, которая смещает объекты файла MIF/MID на заданные приращения координат ΔX и ΔY:

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

#define NKEYS 12
#define STRMAX 1024

/* ========================================================================== *
 * shiftxy
 *
 * Программа считывает объекты из одного файла MIF,
 * сдвигает их на dx, dy в координатных единицах слоя и
 * записывает в другой файл MIF.
 *
 * Usage: shiftxy <input> <output> <dx> <dy>
 *
 * Действия с файлами MID не выполняются,
 * просто используйте оригинал после копирования или переименования.
 * ---------------------------------------------------------------------------- */
int main(int argc, char *argv[])
{
  char buf[STRMAX], w[8][128];
  char *keyword[NKEYS] = {
    "Arc", "Ellipse", "Line", "Pline", "Point", "Region", "Rect",
    "Roundrect", "Text", "Multipoint", "Center", "Multiple"
  };
  double dx, dy, x, y, x2, y2;
  int nsec, npt, t, k;
  FILE *fp0, *fp1;

  if (argc < 5) {
    puts("usage: shiftxy <input> <output> <dx> <dy>");
    exit(EXIT_SUCCESS);
  }
  if ((fp0 = fopen(argv[1], "r")) == NULL) {
    fprintf(stderr, "can't open %s\n", argv[1]);
    exit(EXIT_FAILURE);
  }
  if ((fp1 = fopen(argv[2], "w")) == NULL) {
```

```
  fprintf(stderr, "can't create %s\n", argv[2]);
  exit(EXIT_FAILURE);
}
dx = atof(argv[3]);
dy = atof(argv[4]);
nsec = npt = t = 0;
while (fgets(buf, STRMAX, fp0) != NULL) {
  if (npt == 0) {
    if (nsec == 0) {
      switch (t) {
      case 2: /* Text string */
        fputs(buf, fp1);
        t = 1;
        break;
      case 1: /* Text bbox */
        sscanf(buf, "%s %s %s %s", w[0], w[1], w[2], w[3]);
        x = atof(w[0]) + dx;
        y = atof(w[1]) + dy;
        x2 = atof(w[2]) + dx;
        y2 = atof(w[3]) + dy;
        fprintf(fp1, "    %f %f %f %f\n", x, y, x2, y2);
        t = 0;
        break;
      default:
        sscanf(buf, "%s %s %s %s %s %s %s",
               w[0], w[1], w[2], w[3], w[4], w[5], w[6]);
        for (k = 0; k < NKEYS - 1; k++) {
          if (strcmp(w[0], keyword[k]) == 0)
            break;
        }
        switch (k) {
        case 8: /* "Text" */
          t = 2;
          fputs(buf, fp1);
          break;
        case 9: /* "Multipoint" */
          npt = atoi(w[1]);
          fputs(buf, fp1);
          break;
        case 5: /* "Region" */
          nsec = atoi(w[1]);
          fputs(buf, fp1);
          break;
        case 3: /* "Pline" */
          if (strcmp(w[1], keyword[NKEYS - 1]) == 0) /* "Pline Multiple" */
            nsec = atoi(w[2]);
          else
            npt = atoi(w[1]);
          fputs(buf, fp1);
          break;
        case 4: /* "Point" */
          x = atof(w[1]) + dx;
          y = atof(w[2]) + dy;
          fprintf(fp1, "%s %f %f\n", w[0], x, y);
          break;
        case 10: /* "Center" */
          x = atof(w[1]) + dx;
          y = atof(w[2]) + dy;
          fprintf(fp1, "    %s %f %f\n", w[0], x, y);
          break;
        case 1: /* "Ellipse" */
        case 2: /* "Line" */
        case 6: /* "Rect" */
          x = atof(w[1]) + dx;
          y = atof(w[2]) + dy;
```

```
            x2 = atof(w[3]) + dx;
            y2 = atof(w[4]) + dy;
            fprintf(fp1, "%s %f %f %f %f\n",
                    w[0], x, y, x2, y2);
            break;
          case 7: /* "Roundrect" */
            x = atof(w[1]) + dx;
            y = atof(w[2]) + dy;
            x2 = atof(w[3]) + dx;
            y2 = atof(w[4]) + dy;
            fprintf(fp1, "%s %f %f %f %f %s\n",
                    w[0], x, y, x2, y2, w[5]);
            break;
          case 0: /* "Arc" */
            x = atof(w[1]) + dx;
            y = atof(w[2]) + dy;
            x2 = atof(w[3]) + dx;
            y2 = atof(w[4]) + dy;
            fprintf(fp1, "%s %f %f %f %f %s %s\n",
                    w[0], x, y, x2, y2, w[5], w[6]);
            break;
          default:
            fputs(buf, fp1);
          }
        }
      } else {
        npt = atoi(buf);
        fputs(buf, fp1);
        nsec--;
      }
    } else {
      sscanf(buf, "%s %s", w[0], w[1]);
      x = atof(w[0]) + dx;
      y = atof(w[1]) + dy;
      fprintf(fp1, "%f %f\n", x, y);
      npt--;
    }
  }
  fclose(fp1);
  fclose(fp0);
  return 0;
}
/* ============================================================================ */
```

Сохраним код в файл **shiftxy.c**. Исполняемый модуль можно создать, например, компилятором **gcc**:

```
$ gcc -o shiftxy shiftxy.c
```

Для MS Windows можно загрузить уже скомпилированную программу.

Утилита **shiftxy** запускается в командной строке с четырьмя аргументами: имя входного файла MIF, имя выходного файла MIF, сдвиг Δ*X*, сдвиг Δ*Y*. Переместим объекты **GSHHS_c_L1_1.MIF** на 360 координатных единиц слоя к западу и запишем в **GSHHS_c_L1_2.MIF**:

```
shiftxy GSHHS_c_L1_1.MIF GSHHS_c_L1_2.MIF -360 0
```

Создадим файл **GSHHS_c_L1_2.MID** из **GSHHS_c_L1_2.MID** копированием или переименованием:

```
copy GSHHS_c_L1_1.MID GSHHS_c_L1_2.MID
```

Теперь понятно, зачем мы экспортировали в MIF вместо сохранения в нормальный слой.

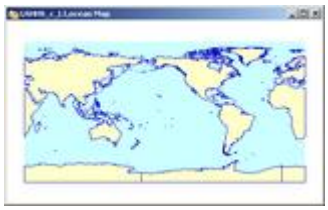
Воссоединение слоя

Импортируем новый MIF с передвинутыми объектами и сольём его с новым слоем **GSHHS_c_L1**:

Откроем новый слой **GSHHS_c_L1**

Импортируем **GSHHS_c_L1_2.MIF** в **GSHHS_c_L1_2**

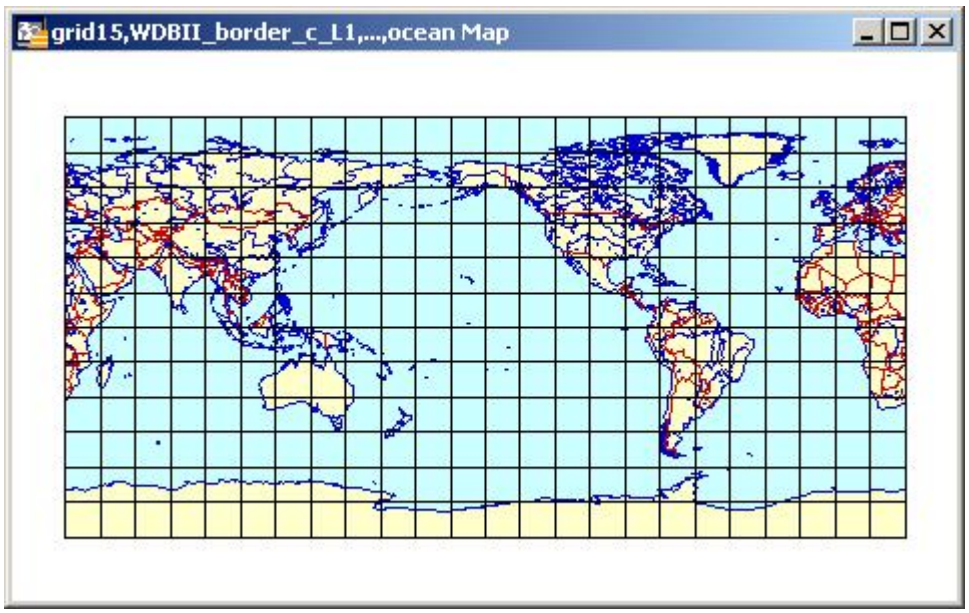
Добавим записи и объекты **GSHHS_c_L1_2** к **GSHHS_c_L1**



Слой **GSHHS_c_L1** со средним меридианом 150° з. д. готов

Отображение карты в проекциях

Повторим описанные действия с несколькими слоями GSHHG, содержащими водоёмы, реки и сухопутные границы, чтобы получить в результате готовую карту.

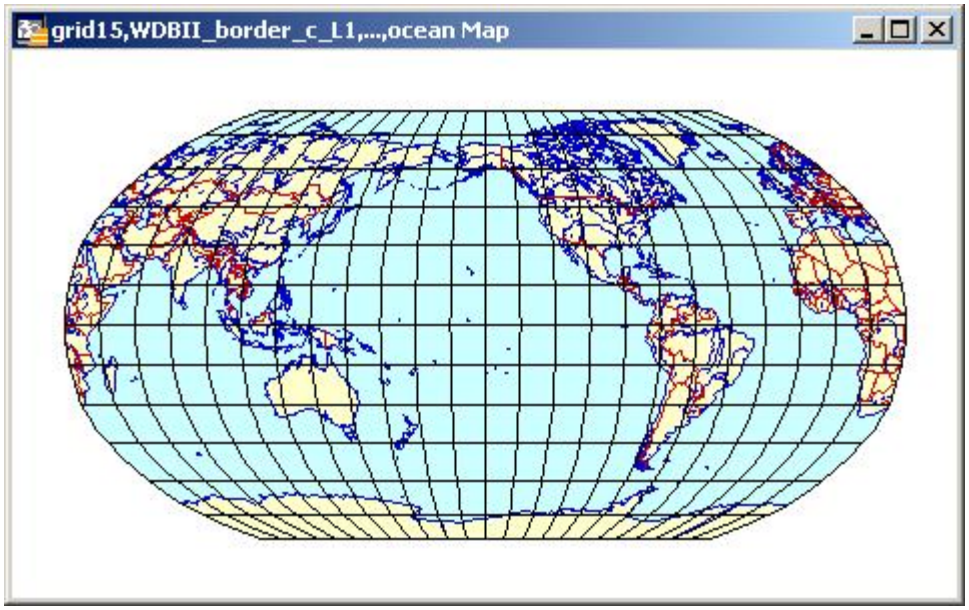


Карта из преобразованных слоёв

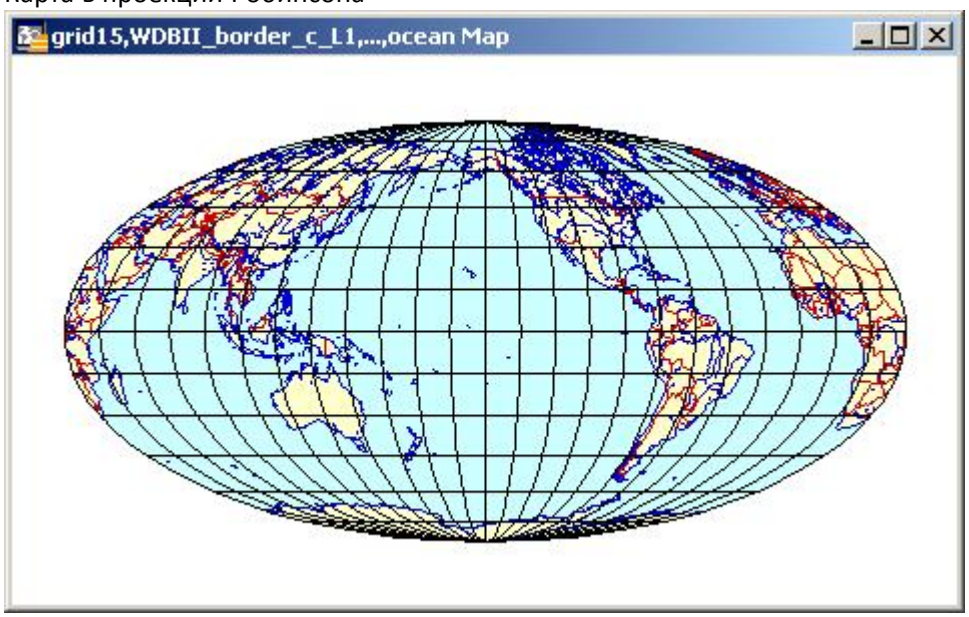
Добавим в файл **MAPINFOW.PRJ** описание проекций Мольвейде и Робинсона с центральным меридианом 150° з. д.:

```
"Mollweide (Equal-Area) 150W WGS84", 13, 104, 7, -150
"Robinson 150W WGS84", 12, 104, 7, -150
```

Отобразим полученные слои в заданных проекциях:



Карта в проекции Робинсона



Карта в проекции Мольвейде

Заключение

Предложенная последовательность действий позволяет получить набор слоёв со средним меридианом, совпадающим с центральным меридианом проекции. Можно усложнить задачу, заменив меридиан сворачивания более сложной линией.

Если подобные задачи возникают часто и связаны с данными больших объёмов, имеет смысл автоматизировать процесс, скажем, написав программу на языке C++ с MapInfo API.

Ссылки

- MapInfo Professional Documentation
- GSHHG — A Global Self-consistent, Hierarchical, High-resolution Geography Database, Paul Wessel, Walter H. F. Smith
- PacWorld

Обсудить в форуме Комментариев — 4

Последнее обновление: 2014-05-15 01:46

Дата создания: 24.03.2013
Автор(ы): ErnieBoyd