```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

struct card {
  char *face;
  char *suit;
};

int main(void) {
  setlocale(LC_ALL, "RU");

  return EXIT_SUCCESS;
}
```

```c
struct employee {
  char firstName[SIZE];
  char lastName[SIZE];
  unsigned int age;
  char gender;
  double hourlySalary;
};
```

```c
  struct employee teamLeader;
  struct employee *teamLeaderPtr;
```

```c
struct card aCard, deck[CARDS], *cardPtr;
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

#define CARDS 52

struct card {
  char *face;
  char *suit;
};

int main(void) {
  setlocale(LC_ALL, "RU");
  struct card deck[CARDS], *cardPtr;
  struct card aCard = { "Тройка", "Червей" };
  printf("%s\n", aCard.face);

  return EXIT_SUCCESS;
}
```

```c
  aCard.face = "Туз";
  aCard.suit = "пик";

  struct card *cardPtr = &aCard;

  printf("%s %s\n%s %s\n%s %s\n",
    aCard.face, aCard.suit,
    cardPtr->face, cardPtr->suit,
    (*cardPtr).face, (*cardPtr).suit);
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

#define CARDS 52

typedef struct {
  char *face;
  char *suit;
} Card;

int main(void) {
  setlocale(LC_ALL, "RU");

  Card deck[CARDS];

  return EXIT_SUCCESS;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <time.h>

#define CARDS 52
#define FACES 13

struct card {
  const char *face;
  const char *suit;
};

typedef struct card Card;

void FillDeck(Card *const deck, const char *face[], const char *suit[]);
void Shuffle(Card *const deck);
void Deal(const Card *const deck);

int main(void) {
  setlocale(LC_ALL, "RU");

  Card deck[CARDS];

  const char *face[] = { "Туз", "Двойка", "Тройка", "Четверка", "Пятерка",
                         "Шестерка", "Семерка", "Восьмерка", "Девятка", "Десятка",
                         "Валет", "Дама", "Король" };
  const char *suit[] = { "червей", "бубен", "треф", "пик" };

  srand(time(NULL));

  FillDeck(deck, face, suit);
  Shuffle(deck);
  Deal(deck);

  return EXIT_SUCCESS;
}
```

```
void FillDeck(Card *const deck, const char *face[], const char *suit[]) {
  for (size_t i = 0; i < CARDS; ++i) {
    deck[i].face = face[i % FACES];
    deck[i].suit = suit[i / FACES];
  }
}

void Shuffle(Card *const deck) {
  for (size_t i = 0; i < CARDS; ++i) {
    size_t j = rand() % CARDS;
    Card temp = deck[i];
    deck[i] = deck[j];
    deck[j] = temp;
  }
}

void Deal(const Card *const deck) {
  for (size_t i = 0; i < CARDS; ++i) {
    printf("%9s %-6s%s", deck[i].face, deck[i].suit,
    (i + 1) % 4 ? " " : "\n");
  }
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

union number {
  int x;
  double y;
};

int main(void) {
  setlocale(LC_ALL, "RU");

  union number value = { 10 };
  //union number value = { 1.43 };

  printf("%d\n", value.x);

  return EXIT_SUCCESS;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

union number {
  int x;
  double y;
};

int main(void) {
  setlocale(LC_ALL, "RU");

  union number value;

  value.x = 100;

  printf("%s\n%s\n%s\n %d\n\n%s\n %f\n\n\n",
         "Записываем 100 в целочисленный элемент",
         "и выводим оба значения.",
         "int:", value.x,
         "double:", value.y);

  value.y = 100.5;

  printf("%s\n%s\n%s\n %d\n\n%s\n %g\n\n\n",
         "Записываем 100.5 в вещественный элемент",
         "и выводим оба значения.",
         "int:", value.x,
         "double:", value.y);

  return EXIT_SUCCESS;
}
```

Битовые операции

Оператор	Наименование	Описание
&	Побитовое И (AND)	Возвращает 1, если оба операнда 1
|	Побитовое ИЛИ (OR)	Возвращает 1, если хотя бы один из операндов 1
^	Исключающее ИЛИ (XOR)	Возвращает 1, если только один из операндов 1
<<	Сдвиг влево	Сдвигает указанное количество бит числа влево, дописывая справа нули
>>	Сдвиг вправо	Сдвигает указанное количество бит числа вправо, дописывая слева нули
~	Дополнение	Меняет значение бита на противоположное

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <limits.h>

void DisplayBits(unsigned int value);

int main(void) {
  setlocale(LC_ALL, "RU");
  unsigned int x;
  printf("%s", "Введите неотрицательное целое значение: ");
  scanf("%u", &x);
  DisplayBits(x);
  return EXIT_SUCCESS;
}

void DisplayBits(unsigned int value) {
  unsigned int displayMask = 1 << (CHAR_BIT * sizeof(unsigned int) - 1);

  printf("%10u = ", value);

  for (unsigned int i = 1; i <= CHAR_BIT * sizeof(unsigned int); ++i) {
    putchar(value & displayMask ? '1' : '0');
    value <<= 1;

    if (i % CHAR_BIT == 0) {
      putchar(' ');
    }
  }

  puts("");
}
```

```
unsigned int number = 65535;
unsigned int mask = 1;
puts("Результат преобразования чисел");
DisplayBits(number);
DisplayBits(mask);
puts("с помощью побитового И (&)");
DisplayBits(number & mask);

number = 15;
unsigned int setBits = 241;
puts("\nРезультат преобразования чисел");
DisplayBits(number);
DisplayBits(setBits);
puts("с помощью побитового ИЛИ (|)");
DisplayBits(number | setBits);

number = 139;
unsigned int secondNumber = 199;
puts("\nРезультат преобразования чисел");
DisplayBits(number);
DisplayBits(secondNumber);
puts("с помощью исключающего ИЛИ (^)");
DisplayBits(number ^ secondNumber);

number = 21845;
puts("\nРезультат преобразования числа");
DisplayBits(number);
puts("с помощью инвертирования (~)");
DisplayBits(~number);
```

```
unsigned int number = 960;
unsigned int shift = 8;

puts("\nРезультат сдвига влево битов числа");
DisplayBits(number);
printf("на %u позиций (<<)\n", shift);
DisplayBits(number << shift);

puts("\nРезультат сдвига вправо битов числа");
DisplayBits(number);
printf("на %u позиций (>>)\n", shift);
DisplayBits(number >> shift);
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

#define CARDS 52

struct bitCard {
  unsigned int face : 4;
  unsigned int suit : 2;
  unsigned int color : 1;
};

typedef struct bitCard Card;

void FillDeck(Card *const deck);
void Deal(const Card *const deck);

int main(void) {
  setlocale(LC_ALL, "RU");

  Card deck[CARDS];

  FillDeck(deck);

  puts("Достоинство карт от 0 до 12 (от туза до короля)");
  puts("Масть карты от 0 до 3 (черви, бубны, трефы, пики)");
  puts("Цвет карты 0 или 1 (красный или черный)\n");

  Deal(deck);

  return EXIT_SUCCESS;
}
```

```
void FillDeck(Card *const deck) {
  for (size_t i = 0; i < CARDS; ++i) {
    deck[i].face = i % (CARDS / 4);
    deck[i].suit = i / (CARDS / 4);
    deck[i].color = i / (CARDS / 2);
  }
}

void Deal(const Card * const deck) {
  printf("%-6s%-6s%-6s%-6s%-6s%s\n",
    "Карта", "Масть", "Цвет",
    "Карта", "Масть", "Цвет");

  for (size_t i = 0, j = i + CARDS / 2; i < CARDS / 2; ++i, ++j) {
    printf("%-6d%-6d%-6d",
    deck[i].face, deck[i].suit, deck[i].color);
    printf("%-6d%-6d%-6d\n",
    deck[j].face, deck[j].suit, deck[j].color);
  }
}
```