Язык C++

Структуры, объединения

Структура

- это одна или несколько переменных (возможно, различных типов), которые для удобства работы с ними сгруппированы под одним именем.

;

Структура

```
struct Point
{
  int x;
  int y;
} pt1, pt2, pt3;

Point p4;
```

- Объявление структуры определяет тип
- Перечисленные в структуре переменные называются *элементами (членами)*

Структура

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

int main(int argc, char* argv[]) {
  Point pt;
  Point max_point = {200,300};

  pt.x  = 200;
  pt.y = 250;

  return 0;
}
```

Вложенные структуры

```
struct Rect  {
  Point pt1;
  Point pt2;
};

int main(int argc, char* argv[]) {
  Point pt;
  pt.x  = 200;
  pt.y = 250;

  Rect rec;
  rec.pt1 = pt;
  rec.pt2.x = 1;
  rec.pt2.y = 2;
  return 0;
}
```

Операции над структурами

- Копирования
- Присваивания
- Взятие адреса
- Доступ к элементам

Операции со структурами

```
Point make_point(int x, int y) {
  Point result;
  result.x = x;
  result.y = y;

  return result;
}

int main(int argc, char* argv[]) {
  Point pt = make_point(239, 1);

  return 0;
}
```

Операции со структурами

```
Point AddPoint(
      Point p1,
      Point p2
) {
  p1.x += p2.x;
  p1.y += p2.y;

  return p1;
}
```

Массивы структур

```
struct Record {
  char name[10];
  char surname[10];
  long phone;
};

Record phonebook[200];
```

Указатели на структуры

```
Record* FindRecord(
    long phone,
    Record* records,
    int count
) {
  for(int i=0; i < count; ++i) {
      if(records[i].phone == phone)
          return &records[i];
  }

  return  nullptr;
}
```

Указатели на структуры

```
Record* key = FindRecord("22345", phonebook, 10);

if(key != nullptr) {
  std::printf("Name: %s  Sername: %s", key->name, key->surname);
}
```

Если ***p*** – указатель на структуру, то ***p->элемент структуры*** ее отдельный элемент

Объединения

- это переменная, которая может содержать (в разные моменты времени) объекты различных типов и размеров. Все требования относительно размеров и выравнивания выполняет компилятор. Объединения позволяют хранить разнородные данные в одной и той же области памяти без включения в программу машинно-зависимой информации.

Объединение

```
union Name {
  struct {
     char name[13];
     char code[3];
  };

  struct {
      int32_t i1;
      int32_t i2;
      int32_t i3;
      int32_t i4;
  };
};
```

Объединение

```
bool NameCompare(const Name& a, const Name& b) {
   return (std::strcmp(a.name, b.name) == 0 && std::strcmp(a.code, b.code)) ;
}

bool IntCompare(const Name& a, const Name& b) {
   return (a.i1 == b.i1 && a.i2 == b.i2 && a.i3 == b.i3 && a.i4 == b.i4);
}
```

Объединение

```
int main() {
    Name a = {.name = "0123456789AB", .code = "12"};
    Name b = {.name = "0123456789AB", .code = "10"};

    const uint64_t retry = 100000000000;

    // …

    return 0;
}
```

Объединение

```
std::chrono::system_clock::time_point begin = std::chrono::system_clock::now();
for(int i = 0; i < retry;  ++i)
     NameCompare(a, b);

std::chrono::system_clock::time_point end = std::chrono::system_clock::now();

std::cout << std::chrono::duration_cast<std::chrono::milliseconds>(end - begin).count()
          << std::endl;
```

Объединение

```
struct Triangle {
  Point a;
  Point b;
  Point c;
};

struct Rect {
  Point left_top;
  Point right_top;
  Point left_bottom;
};
```

Объединение

```
struct Circle {
  Point center;
  float r;
};

enum FigureType{
  Triangle,
  Rect,
  Circle,
};
```

Объединение

```
union FigureU {
  Triangle triangle;
  Rect     rect;
  Circle   circle;
};

struct Figure {
  FigureType type;
  FigureU   fig;
};
```