acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Язык Си» . CoffeStruct :

Структуры

Теория

Массив хранит набор данных одинакового типа. Некоторые объекты нужно описывать данными разного типа.

Например, для описания одного студента нужны строки (имя, фамилия, отчество), целые числа (номер факультета, номер группы, номер в единой системе учета оценок) и массивы (оценки за экзамен в 1 семестре).

Чтобы хранить в одной переменной значения разного типа создают новый тип данных с помощью ключевого слова **struct**.

Объявление нового типа данных

Создадим структуру, которая описывает человека и хранит его рост в сантиметрах (целое число), вес в килограммах, размер обуви и год рождения.

```
struct Human {
    int   height;  // высота в сантиметрах
    float weight;  // вес в килограммах
    int   foot;    // размер обуви
    int   bdate;   // год рождения
};                 // в конце обязательно ТОЧКА С ЗАПЯТОЙ
```

Мы определили новый тип данных **struct Human**.

Новый тип данных называется не Human, не stuct, а **struct Human**. Поэтому в языке с его пишут в два слова. (В языке c++ слово struct при использовании типа можно не писать, в языке с - писать обязательно.)

Объявление переменных и их использование

Создадим переменную x типа int. Присвоим x число 7. Увеличим x на 2.
Создадим переменную alex типа struct Human. И опишем человека ростом 180 см, весом 70.1 кг, с 42 размером обуви 1997 года рождения. После коллоквиума по математике он похудел на 1.24 кг.

```
int x;                  // объявили переменную x типа int
x = 7;                  // переменной x присвоили число 7 (запись в x)
x = x + 2;              // прочитали значение переменной x, вычисли x+2 и результат записали в x

struct Human alex;      // объявили переменную alex типа struct Human

alex.height = 180;      // в поле height переменной alex записали число 180
alex.foot   = 42;       // в поле foot переменной alex записали число 42
alex.bdate  = 1997;     // в поле bdate переменной alex записали число 1997
alex.weight = 70.1;     // в поле weight переменной alex записали число 70.1
                        // после коллоквиума он похудел на 1.24 кг
alex.weight = alex.weight - 1.24;
```

Операция . (точка) дает доступ к полю структуры.

Если переменная типа struct Human, то операция . может дать доступ к полям height, foot, bdate, weight.

Доступ к полю alex.money невозможен. Так как в структуре struct Human нет поля money.

Явная инициализация при объявлении

Данные можно присвоить сразу при объявлении переменной.

Поля идут том же порядке, что и при объявлении типа.

```
struct Human alex = {180, 70.1, 42, 1997};
```

x = y

При присвоении структур идет копирование той области памяти, где расположена переменная y, в область памяти, где расположена x. Это называется побитовым копированием.

Присваивать можно только переменные с одинаковым названием типа.

Переменные разных типов присвоить нельзя, даже если внутри них содержатся такие же данные.

```
struct Human mike;      // объявили новую переменную mike того же типа, что и alex
mike = alex;            // mike тоже теперь весит 70.1 кг, при росте 180 см, нога 42 размера и родился в 1997 году.
```

Структуры в функции передаются по значению

В языке C в функцию все аргументы передаются по значению. Любого типа. Для структур делается побитовая копия.

```
#include <stdio.h>

struct Human {
    int   height;
    float weight;
};

void cut (struct Human h)
{
    h.height = 100;
}
int main()
{
    struct Human alex = {180, 70.1};

    cut(alex);

    printf("height=%d weight=%f\n", alex.height, alex.weight);

    return 0;
}
```

Будет напечатано height=180 width=70.1

В функцию cut в виде параметра h передается копия переменной alex. Изменение этой копии (и уничтожение ее в конце выполнения функции cut) не изменит значение оригинала.

Структура в виде возвращаемого значения

Можно вернуть из функции значение какой-либо структуры. Вернем из функции cut значение struct Human:

```
#include <stdio.h>

struct Human {
    int   height;
    float weight;
};

struct Human cut (struct Human h)
{
    h.height = 100;
}
int main()
{
    struct Human alex = {180, 70.1};
    struct Human mike;

    mike = cut(alex);

    printf("height=%d weight=%f\n", alex.height, alex.weight);
    printf("height=%d weight=%f\n", mike.height, mike.weight);

    return 0;
}
```

Будет напечатано:
height=180 width=70.1
height=100 width=70.1

Указатели на структуры

Как передавать переменную в функцию, чтобы изменять значение внутри функции?

Надо передать функцию значение адреса этой переменной.

```
#include <stdio.h>

void cut (struct Human * adr) {
    // оператор ->, так как у adr тип УКАЗАТЕЛЬ на struct Human
    adr -> height = 100;
}

int main()
{
```

```
    struct Human alex = {180, 70.1};

    cut(& alex);

    // оператор точка, так как у alex тип struct Human
    printf("height=%d weight=%f\n", alex.height, alex.weight);

    return 0;
}
```

Напечатает height=100 weight=70.1

Оператор -> придумали, чтобы не писать (*alex).height.
-> удобнее читать.

Задачи

Отрезок двухмерный (на осях X и Y) (КИТАЙ)

```
#include <stdio.h>

struct Point {
   int x;
   int y;
};

struct Line {
   struct Point start;
   struct Point end;
};
void printPoint( struct Point p)
{
   printf("(%d, %d) ", p.x, p.y);
}
struct Point getPoint( void)
{
   struct Point p;
   scanf("%d%d", & p.x, & p.y);
   return p;
}
struct Point movePointX(struct Point p, int dx)
{
   struct Point res;
   res.x = p.x + dx;
   res.y = p.y;
   return res;
}

void printLine( struct Line lin)
{
   printPoint (lin.start);
   printPoint (lin.end);
   printf("\n");
}
struct Line getLine( void)
{
   struct Line t;
   t.start = getPoint();
   t.end   = getPoint();
   return t;
}
struct Line moveLine(struct Line lin, int dx)
{
   struct Line res;
   res.start = movePointX(lin.start, dx);
   res.end   = movePointX(lin.end  , dx);
   return res;
}
int main()
{
   struct Line a;
   struct Line b;
   int dx;

   a = getLine();
   scanf("%d", &dx);
   printLine(a);

   b = moveLine (a, dx);

   printLine(b);

   return 0;
}
```

Отрезок одномерный (на оси X)

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

```
struct Line {      // название структуры
   int start;      // поля структуры
   int finish;
};
struct Line readLine();             // читает отрезок
void   printLine (struct Line a);   // печатает отрезок
int      lengthLine(struct Line a);   // длина отрезка
struct Line moveLine (struct Line a, int dx); // двигает отрезок на dx

int main() {
   struct Line a, b;
   int dx;
   a = readLine();
   scanf("%d", &dx);

   printf("length = %d\n", lengthLine(a));
   b = moveLine(a, dx);

   printLine(a);
   printLine(b);
   return 0;
}
struct Line readLine() {
   struct Line t;
   scanf("%d%d", &t.start, &t.finish);
   return t;
}
void printLine(struct Line a) {
   printf("(%d, %d)\n", a.start, a.finish);
}
int lengthLine(struct Line a) {
   return a.finish - a.start;
}
struct Line moveLine(struct Line a, int dx) {
   struct Line tmp;
   tmp.start = a.start + dx;
   tmp.finish = a.finish + dx;
   return tmp;
}
```

Отрезок одномерный (с указателями)

```
/* Отрезок на оси X. Научимся его читать, печатать и сдвигать на dx */
#include <stdio.h>

struct Line {      // название структуры
   int start;      // поля структуры
   int finish;
};
void readLine(struct Line * pa);   // читает отрезок
void printLine (struct Line a);      // печатает отрезок
int    lengthLine(struct Line a);      // длина отрезка
void moveLine (struct Line * pa, int dx); // двигает отрезок на dx

int main() {
   struct Line a;
   int dx;
   readLine(&a);
   scanf("%d", &dx);

   printf("length = %d\n", lengthLine(a));
   printLine(a);
   moveLine(&a, dx);
   printLine(a);
   return 0;
}
void readLine(struct Line * pa) {
   scanf("%d%d", &(pa->start), &(pa->finish));
   return t;
}
void printLine(struct Line a) {
   printf("(%d, %d)\n", a.start, a.finish);
}
int lengthLine(struct Line a) {
   return a.finish - a.start;
}
void moveLine(struct Line * pa, int dx) {
   pa->start = pa->start + dx;
   pa->finish += dx;
}
```

Отрезок на плоскости XY

```
/* Отрезок на плоскости XY. Научимся его читать, печатать и сдвигать
   Какой отрезок больше? */
#include <stdio.h>
#include <math.h>
struct Direction {   // для описания сдвига
   int x;
```

```
   int y;
};
struct Point {
   int x;
   int y;
};
struct Line {      // название структуры
   struct Point start;      // поля структуры
   struct Point finish;
};
struct Line readLine();              // читает отрезок
void   printLine (struct Line a);    // печатает отрезок
float  lengthLine(struct Line a);    // длина отрезка
struct Line moveLine (struct Line a, int dx); // двигает отрезок на dx

int main() {
   struct Line a, b;
   int dx;
   a = readLine();
   scanf("%d", &dx);

   printf("length = %f\n", lengthLine(a));
   b = moveLine(a, dx);

   printLine(a);
   printLine(b);
   return 0;
}
struct Point readPoint() {
   struct Point t;
   scanf("%d%d", &t.x, &t.y);
   return t;
}

struct Line readLine() {
   struct Line t;
   struct Point a, b;
   a = readPoint();
   b = readPoint();
   t.start = a;
   t.finish = b;
   return t;
}
void printPoint(struct Point a) {
   printf("(%d, %d) ", a.x, a.y);
}
void printLine(struct Line lin) {
   printPoint(lin.start);
   printPoint(lin.finish);
   printf("\n");
}
float lengthLine(struct Line lin) {
   int x = lin.start.x - lin.finish.x;
   int y = lin.start.y - lin.finish.y;
   float length = sqrt(x*x + y*y);
   return length;
}
struct Line moveLine(struct Line a, int dx) {
   struct Line tmp = a;
   tmp.start.x += dx;
   tmp.finish.x += dx;
   return tmp;
}
```

Прямоугольник на плоскости

```
/* Прямоугольник на плоскости XY. Научимся его читать, печатать и сдвигать  */
#include <stdio.h>
struct Direction {
   int x;
   int y;
};
struct Point {
   int x;
   int y;
};
struct Rect {      // название структуры
   struct Point lt;      // поля структуры
   struct Point rb;
};
struct Direction readDirection() {
   struct Direction t;
   scanf("%d%d", &t.x, &t.y);
   return t;
}

struct Point readPoint() {
   struct Point t;
   scanf("%d%d", &t.x, &t.y);
   return t;
```

```
}
void printPoint(struct Point a) {
   printf("(%d, %d) ", a.x, a.y);
}

struct Rect readRect() {
   struct Rect t;
   struct Point a, b;
   a = readPoint();
   b = readPoint();
   t.lt = a;
   t.rb = b;
   return t;
}
void printRect(struct Rect rect) {
   printPoint(rect.lt);
   printPoint(rect.rb);
   printf("\n");
}
void moveRect(struct Rect * p, struct Direction dir)
{
   p->lt.x = p->lt.x + dir.x;
   p->lt.y = p->lt.y + dir.y;
   p->rb.x += dir.x;
   p->rb.y += dir.y;
}
int main()
{
   struct Rect rect;
   struct Direction d;
   rect = readRect();
   d = readDirection();

   moveRect(&rect, d);
   printRect(rect);

   return 0;
}
```

Еще теории

Декларации

Анонимные структуры

union

Битовые поля

-- TatyanaDerbysheva - 07 Nov 2014

(c) Материалы раздела "Язык Си" публикуются под лицензией GNU Free Documentation License.