**10.04.2024 года Структуры как тип пользователя**

**Определение структуры, структурной переменной. Вложенные структуры, массивы в структурах, массив структур. Структуры и указатели. Динамические структуры, массивы структур. Ссылка на структуру. Структуры и функции.**

**Определение структуры, структурной переменной**

**Структура** – это тип **пользователя**, объединяющий под **одним** именем данные ***одного или разных*** типов, называемые ***полями***(*элементами*)***структуры***. В качестве полей могут быть *переменные простых типов, массивы, другие структуры, указатели* и т. д. В более поздних версиях языка С++ в структуру можно включать и функции (методы). Структуры позволяют сгруппировать данные, объединяемые каким-либо общим понятием. Например, имя, год и месяц рождения:

struct **Date**{ // **Date**– имя нового ***типа*** пользователя

char name[10]; // поле **name**

int year; // поле **year**

int month; // поле **month**

};

**Определение структурной переменной.** После определения **struct-**типа***,*** можно объявить *переменную структурного типа*, причём двумя способами:

**Date** st; // переменная **st** типа **Date**

или

struct **Date**{

char name[10];

int year;

int month;

}**st**;

// **Пример 1.** Определение структуры, структурной переменной.

// Доступ к полям --- используется ***операция точка*** (.).

// Инициализация полей структуры.

// Операции над структурными переменными.

// !! **Имя struct-типа ---- c большой буквы**!!

#include <iostream>

using namespace std;

struct **Date**{ // **Date** – имя нового типа пользователя

char name[10]; // поле name

int year; // поле year

int month; // поле month

};

**int main()** {

bool fl;

**Date** st1; // структурная переменная **st1** типа **Date**

cout << "Enter name: "; cin >> st1.name; // ввод с клавиатуры

st1.year = 2000; // присваивание

st1.month = 11;

cout << "st1: " << st1.name << " " << st1.year << " " << st1.month << endl;

**Date** st2 = { "Ivanov", 1999, 5 }; // явная инициализация полей

cout << "st2: " << st2.name << " " << st2.year << " " << st2.month << endl;

st2 = st1; // присваивание структур, поля **st1** **побитово**

// копируются в **st2. Один тип у структур!!!**

cout << "st2: " << st2.name << " " << st2.year << " " << st2.month << endl;

// **if(st1 == st2) оператор;** // **Нельзя** – ошибка компиляции!

if((st1.year == st2.year) && (st1.month == st2.month)) fl = true;

else fl = false;

cout << "fl = " << fl << endl;

}

**Результат:**

Enter name: **Petrov**

st1: **Petrov** 2000 11

st2**: Ivanov** 1999 5 st2: **Petrov** 2000 11 fl = 1

Определение **структуры** задаёт её внутреннюю организацию, описывая поля, входящие в её состав.

Определение **структурной переменной** означает, что ей выделяется память. В частности, для **st1***,* **st2** резервируется память по **18** байтов для каждой.

**Вложенные структуры, массивы в структурах, массив структур**

Если полем структуры является другая структура, то получается *вложенная структура*. Доступ к полям вложенных структур осуществляется с помощью нескольких *операций точка* **(.).**

Полем структуры может быть одномерный или двумерный массивы фиксированной размерности.

Для объявления ***массива структур*** сначала необходимо определить структурный тип, а затем объявить массив данного типа.

// **Пример 2.** Вложенные структуры. Массивы в структурах.

// Массив структур.

#include <iostream>

using namespace std;

struct **Date** { // тип **Date**

int day, month, year;

};

struct **Time** { // тип **Time**

int hour, min, sec;

};

int **main()** {

const int n = 20;

struct **DateTime** {

**Date** today; // вложенная структура

**Time** clock ; // вложенная структура

}**dt;**  // **dt** – переменная

dt.today.year = 2001; // доступ к полям **вложенной**

dt.clock.min = 33; // структуры

cout << dt.today.year << " " << dt.clock.min << endl; // **2001 33**

// Массивы в структурах

struct **Array**{ // тип **Array**

int matr[2][3]; // поля структуры – массивы

float sol[3];

char fio[n];

}**ar;**

ar.matr[0][0] = 25; // доступ к полям

ar.sol[2] = 5.5;

ar.fio[0] = **'\***';

cout << ar. matr[0][0] <<' '<< ar.sol[2] <<' ' <<ar.fio[0] << endl; // **25 5.5 \***

**// Массив структур**

struct **Person** { // тип **Person**

char fam[n];

Date birthday;

long telNumber;

};

Array mas[n]; // массив структур **mas** типа **Array**

Date list[10]; // массив структур **list** типа **Date**

Person per[n]; // массив структур **per** типа **Person**

mas[0].matr[0][1] = 25;

mas[1].sol[0] = 15.5;

cout << mas[0].matr[0][1] << ' ' << mas[1].sol[0] << endl; // **25 15.5**

list[1].day = 25;

**// list.day[1] = 25;**  // **нельзя!!!**

per[1].birthday.day = 5;

cout << list[1].day<<' '<<per[1].birthday.day<<endl; // **25 5**

}

**Структуры и указатели**

**Указатель как поле структуры**. Полями структуры наряду с другими переменными могут быть и *указатели*.

Если поле структуры указатель, его необходимо **проинициализировать** перед использованием.

// **Пример 3.** Указатели в структуре.

#include <iostream>

using namespace std;

struct **Rab** { // тип **Rab**

int \* px;

int y;

};

int **main()**{

**Rab** st;

int z; st.px = &z; // инициализация указателя **px**

\*st.px = 5; // **символ \*** относится к полю **px**

int u = \*st.px;

st.y = 77;

cout << \*st.px << ' ' << st.y << ' ' << u << ' ' << z << endl; // **5 77 5 5**

z = 6;

cout << \*st.px << ' ' << st.y << ' ' << u << ' ' << z << endl; // **6 77 5 6**

st.px = new int(4); // **st.px** -- адрес динамической переменной

cout << \*st.px << ' ' << st.y << ' ' << u << ' ' << z << endl; // **4 77 5 6**

delete st.px;

int k;

cout<<"Enter size array: "; cin>>k; // например, к = 4

st.px = new int[k]; // **st.px** -- адрес динамического массива

for (int i = 0; i < k; i++){

st.px[i] = i + 1;

cout << st.px[i] << ' ';

cout << \*(st.px+i) <<' ' ; // **1 1 2 2 3 3 4 4**

**}**

cout<<endl;

cout << st.px << endl; // **014E4A90 (адрес)**

delete [] st.px;

}

**Указатель на структуру**. Можно объявить, используя обычный синтаксис, указав тип --- структура:

Rab \* uk;

Перед использованием указателя на структуру его необходимо **проинициализировать**.

Для ***доступа*** к полям структурычерез ***указатель на структуру*** имеется две возможности:

* вместо *операции* *точка* **(.)** использовать *операцию стрелка* **(->)**;
* выполнить *разыменование указателя* и использовать уточнённое имя поля структуры с помощью *операции точка* **(.)**.

// **Пример 4.** Указатель на структуру.

// Операция стрелка (->) для доступа к полям структуры.

// Операция разыменования указателя **(\*)** и операция точка **(.)** для доступа

// к полям структуры.

#include <iostream>

using namespace std;

struct Rab { // тип **Rab**

int\* px;

int y;

};

int **main()**{

// Использование **операции стрелка** **(->**) для доступа к полям структуры

Rab st; Rab \* ptr = &st;

ptr->y = **777**;

int z;

ptr->px = &z;

\*ptr->px = **555**; // символ **\*** относится к полю **px**

cout<<ptr->y<<' '<<\*ptr->px << ' ' <<z <<endl; // **777 555 555**

// Использование операции **разыменования указателя** **(\*)** и **операции точка** **(.)**

// для доступа к полям структуры

(\*ptr).y **= 888**; // символ **\*** относится к указателю **ptr**

(\*ptr).px = &z; \*((\*ptr).px) = **666**;

cout << (\*ptr).y << ' ' << \*((\*ptr).px) << ' ' << z << endl; // **888 666 666**

}

**Динамические структуры, массивы структур**

Используя *операцию* **new** и указатель на структуру, можно создать динамическую структуру и динамический массив структур:

Rab \* ptr = new Rab;

Rab \* p = new Rab[3];

Для освобождения динамической памяти применяется *операция* **delete**:

delete ptr; // **delete** динамическая переменная

delete [] p; // **delete** динамический массив

// **Пример 5**. Динамические структуры, массивы структур.

#include <iostream>

#include <cstring>

using namespace std;

struct Rab { // тип **Rab**

int \* px;

int y;

};

int **main()** {

const int buf = 5;

// Динамические структуры

int z;

Rab\* p = new Rab; // указателю **p** присваивается адрес структуры

p->y = 88;

p->px = &z; \*p->px = 66; // или **\*((\*p).px) = 66;**

cout << p->y << ' ' << \*p->px << ' ' << z << endl; // **88 66 66**

delete p;

// Динамический массив структур

Rab \* q = new Rab[2]; // динамический массив структур

cout**<<**"Array q --- operation point (.):"<<endl;

for (int i = 0; i < 2; i++) {

q[i].y = i;

q[i].px = &z; \*q[i].px = i;

cout << q[i].y << ' ' << \*q[i].px << ' ' << z << endl; // **0 0 0 1 1 1**

}

cout**<<**"Array q --- operation arrow (->):"<<endl;

for (int i = 0; i < 2; i++) {

(q+i)->y = i+1;

(q + i)->px = &z; \*(q+i)->px = i+1;

cout << (q + i)->y << ' ' << \*(q + i)->px << ' ' << z << endl; // **1 1 1 2 2 2**

**}**

delete [] q;

}

**Ссылка на структуру**

Для структур, как и для простых переменных, можно использовать ссылочный тип. Например:

struct **Rac**{

float a;

float b;

}st;

Rac & rst = st;

После такого объявления **st** и **rst** – это два синонимa одной и той же переменной структурного типа **Rac**. Поэтому, если изменить поле **x** или **y** структуры **st**, то изменится соответствующее поле структуры **rst**.

**Структуры и функции**

**Передача в функцию структуры.** Параметром функции может быть *структура*. При вызове такой функции тип структуры, используемой в качестве аргумента, должен соответствовать типу параметра. Для этого, как правило, структурный тип определяется ***глобально****,* а затем используется имя структурного типа, как при определении функции, так и при её вызове.

**// Пример 6.** Определить структуру, полями которой являются числитель

// и знаменатель рациональной дроби. Создать функции для сложения

// двух дробей. Структуры передаются в функции **по значению**.

// Функция **sumStruct\_1** возвращает **число**.

// Функция **sumStruct\_2** возвращает **структуру**.

#include <iostream>

using namespace std;

struct **Rac**{

double a;

double b;

};

double **sumStruct\_1**(Rac s1, Rac s2); // прототип **sumStruct()\_1**

Rac **sumStruct\_2**(Rac s1, Rac s2); // прототип **sumStruct()\_2**

int **main()**{

Rac st1, st2; // переменные **st1, st2** типа **Rac**

cout<<"Enter a, b for st1: ";

cin>>st1.a>>st1.b; // **1 2**

cout<<"Enter a, b for st2: ";

cin>>st2.a>>st2.b; // **3 4**

double rez = **sumStruct\_1**(st1, st2);

cout<<"rez = "<<rez<<endl; // rez = **1.25**

Rac rez1 = **sumStruct\_2**(st1, st2);

cout<<"rez1 = "<<rez1.a<<'/'<<rez1.b<<endl; // rez1 = **10/8**

}

double **sumStruct\_1** (Rac s1, Rac s2){ // функция **sumStruct()\_1**

double z = ((s1.a \* s2.b) + (s1.b \* s2.a)) / (s1.b \* s2.b);

return z;

}

Rac **sumStruct\_2**(Rac s1, Rac s2){ // функция **sumStruct()\_2**

Rac t;

t.a = (s1.a \* s2.b) + (s1.b \* s2.a);

t.b = s1.b \* s2.b;

return t;

}

// **Пример 7.** Определить структуру для работы с рациональной дробью.

// Создать функцию для сложения двух дробей.

// Структуры передаются в функцию **по значению**. Результат возвращается

// через **параметр-указатель** и через **параметр-ссылку** на структуру.

#include <iostream>

using namespace std;

struct **Rac**{

double a;

double b;

};

void **sumStruct**(Rac s1, Rac s2, Rac\* s3, Rac **& s4**); // прототип **sumStruc()**

int **main(){**

Rac st1, st2, rez, rez1; // переменные **st1, st2**

cout<<"Enter a, b for st1: ";

cin>>st1.a>>st1.b; // **1 2**

cout<<"Enter a, b for st2: ";

cin>>st2.a>>st2.b; // **3 4**

sumStruct(st1, st2, &rez, rez1);

cout<<"rez = "<<rez.a<<'/'<<rez.b<<endl; // **rez** = **10 / 8**

cout<<"rez1 = "<<rez1.a<<'/'<<rez1.b<<endl; // **rez1** = **10 / 8**

}

void **sumStruct**(Rac s1, Rac s2, Rac\* s3, Rac **& s4**){

// (\*s3).a = (s1.a \* s2.b) + (s1.b \* s2.a);

// (\*s3).b = s1.b \* s2.b;

s3->a = (s1.a \* s2.b) + (s1.b \* s2.a); // через указатель

s3->b = s1.b \* s2.b;

s4.a = (s1.a \* s2.b) + (s1.b \* s2.a); // через ссылку

s4.b = s1.b \* s2.b;

}

**Передача в функцию массива структур**. В функцию можно также передавать адреc массива структур аналогичным образом, как это мы делали для числовых массивов, что и будет показано в следующем примере.

// **Пример 8.** Создание динамического массива структур, каждый элемент

// которого содержит информацию о студенте:

// фамилия (вводим); массив оценок (вводим); средний балл (вычисляем).

// Передача адреса массива в функции **input(), show(), middleMark()**.

#include <iostream>

using namespace std;

#include <iomanip>

const int m = 4, k = 10;

struct **Stud**{ // описание структурного типа **Stud**

char name[k]; // можно char \* name;

int mark[m];

float middle;

};

void input(Stud\*, int, int); // прототип **input()**

void show(Stud\*, int); // прототип **show()**

float midMark(int\*, int); // прототип **midMark()**

int **main()**{

int n;

cout << "Enter number student : ";

cin >> n; // количество студентов

**Stud**\* array = new Stud[n]; // **динамический** массив структур

input(array, n, m);

show(array, n);

delete [] array;

}

void **input**(Stud\* mas, int size1, int size2) { // функция **input()**

for (int i = 0; i < size1; i++) {

// если char \* name; то mas[i].name = new char[k];

cout << "Enter name: ";

cin >> mas[i].name; // ввод имени

cout << "Enter 4 mark: ";

for (int j = 0; j < size2; j++)

cin >> mas[i].mark[j]; // ввод оценок

mas[i].middle = midMark(mas[i].mark, size2);

}

}

void **show**(Stud\* mas, int size){ // функция **show()**

cout << "Spisok student" << endl;

cout << "Name: Middle mark:"<< endl;

for (int i = 0; i < size; i++)

cout << setw(10) << left << mas[i].name << mas[i].middle << endl;

}

float **midMark**(int\* mas, int size) { // функция **midMark()**

float sum = 0, sr;

for (int i = 0; i < size; i++)

sum = sum + mas[i];

sr = sum/size; // средний балл

return sr;

}

**Результат:**

Enter number student: 2

Enter **name**: aaaa

Enter 4 mark: 4 5 6 7

Enter **name**: bbbb

Enter 4 mark: 7 6 7 6

**Spisok student**

**Name:** **Middle mark**:

aaaa 5.5

bbbb 6.5

// **Пример 9.** Создать **статический** массив структур, каждый , каждый элемент

// которого содержит информацию о студенте: фамилия, имя, отчество (вводим);

// массив оценок (вводим); средний балл (вычисляем).

// Передача адреса массива функциям **input(), show(), middleMark()**.

#include <iostream>

using namespace std;

#include <iomanip>

const int m = 4, k = 10;

struct **Student**{ // описание структурного типа **Student**

char name[k];

int mark[m];

float middle;

};

void input(Student\*, int, int); // прототип **input()**

void show(Student\*, int); // прототип **show()**

float midMark(int\*, int); // прототип **midMark()**

int **main()**{

const int n = 3; // количество студентов

Student array[n]; // **статический** массив структур

input(array, n, m);

show(array, n);

}….