# Обработка на структури и масиви от структури

Структура се използва като термин за създаване на потребителски дефиниран тип данни. Структурата групира различни типове данни, с цел да се опише характеристиката на предмет(обект). Крайният резултат е наш тип данни, който може да ползваме с познатите операции в езика, но обединява цялата информация за описания обект в един логически елемент. По този начин можем да улесним работата с по сложни логически реализации и да опростим писането на код за комплексни модели от информация.

Трябва да се прави разлика между структура и структура от данни. Втория термин дефинира правилата за ползване на информация като внася алгоритъм и правила за обработка на данните, които съхраняваме.

Пример за ползване на структура:

Нека предположим, че искаме да направим списък с книгите, които имаме в библиотеката ни. Всяка книга се характеризира с определени елементи, уникални за нея например:

* Заглавие
* Име на автор
* Тема
* Идентификационен номер

Понеже всичко това характеризира една книга, има логика ако искаме да направим елемент от вид книга да дефинираме структура, която да обедини тези характеристики на едно място.

# Дефиниция

За дефиниция на структура използваме ключовата дума “struct”. След това следва посочване на името на структурата, обикновено свързано с предмета, който описваме. В тялото на структурата се описват полетата, които трябва да опишат характеристиката на тази структура. Те могат да бъдат от всеки обявен до момента тип достъпен в езика.

Структура се дефинира в общото пространство на програмата извън функции!!!

***struct Book***

***{***

***char title[20];***

***char author[20];***

***char subject[200];***

***int book\_id;***

***} <променливи>;***

Така създадената структура има 4 променливи, три от които са низове, и едно цяло число. Когато обявяваме променливи от тип масив трябва задължително да посочим дължината им.

След затварящата скоба на структурата може да напишем имена на променливи, които ще бъдат създадени като глобални и може да се ползват в цялата програма.

Всяка структура приключва със знака ‚ ; ‘.

# Създаване и достъп до елементите на структура

Елемент от вид структурата създаваме като напишем името на структурата и името на променливата:

***struct Book book1;***

Когато искаме да достъпим поле от структурата използваме ‚ . ‘:

***book1.title***

пример:

#include <stdio.h>

#include <string.h>

struct Books {

char title[50];

char author[50];

char subject[100];

int book\_id;

};

int main( ) {

struct Books Book1;

struct Books Book2;

strcpy( Book1.title, "C Programming");

strcpy( Book1.author, "Spiridon Haralampiev");

strcpy( Book1.subject, "C Programming Tutorial");

Book1.book\_id = 6495407;

/\* book 2 specification \*/

strcpy( Book2.title, "BPE materials");

strcpy( Book2.author, "Haralampi Spiridonov");

strcpy( Book2.subject, "Materials for BPE subject");

Book2.book\_id = 6495700;

/\* print Book1 info \*/

printf( "Book 1 title : %s\n", Book1.title);

printf( "Book 1 author : %s\n", Book1.author);

printf( "Book 1 subject : %s\n", Book1.subject);

printf( "Book 1 book\_id : %d\n", Book1.book\_id);

/\* print Book2 info \*/

printf( "Book 2 title : %s\n", Book2.title);

printf( "Book 2 author : %s\n", Book2.author);

printf( "Book 2 subject : %s\n", Book2.subject);

printf( "Book 2 book\_id : %d\n", Book2.book\_id);

return 0;

}

# Работа с функции

Използвайки функции работим по същият начин, по който боравим с обикновена променлива или указател.

/\* function declaration \*/

void printBook( struct Books book );

…

Горният код

….

/\* print Book1 info \*/

printBook( Book1 );

/\* Print Book2 info \*/

printBook( Book2 );

return 0;

}

void printBook( struct Books book ) {

printf( "Book title : %s\n", book.title);

printf( "Book author : %s\n", book.author);

printf( "Book subject : %s\n", book.subject);

printf( "Book book\_id : %d\n", book.book\_id);

}

# Указател към структура

Използването на указатели към структурата не се различава от нормалното ползване на указател, с малка особеност при влизането към полетата от структурата:

struct Books \*struct\_pointer;

взимане на адреса на реална променлива:

struct\_pointer = &Book1;

достъп до поле от указателя:

struct\_pointer->title;

горната програма чрез структури:

#include <stdio.h>

#include <string.h>

struct Books {

char title[50];

char author[50];

char subject[100];

int book\_id;

};

/\* function declaration \*/

void printBook( struct Books \*book );

int main( ) {

struct Books Book1;

struct Books Book2;

strcpy( Book1.title, "C Programming");

strcpy( Book1.author, "Spiridon Haralampiev");

strcpy( Book1.subject, "C Programming Tutorial");

Book1.book\_id = 6495407;

/\* book 2 specification \*/

strcpy( Book2.title, "BPE materials");

strcpy( Book2.author, "Haralampi Spiridonov");

strcpy( Book2.subject, "Materials for BPE subject");

Book2.book\_id = 6495700;

/\* print Book1 info by passing address of Book1 \*/

printBook( &Book1 );

/\* print Book2 info by passing address of Book2 \*/

printBook( &Book2 );

return 0;

}

void printBook( struct Books \*book ) {

printf( "Book title : %s\n", book->title);

printf( "Book author : %s\n", book->author);

printf( "Book subject : %s\n", book->subject);

printf( "Book book\_id : %d\n", book->book\_id);

}

# Битови полета

Особеност е ползването на полета, като дефинираме точно колко бита искаме да е променливата. По този начин може да имаме структура, която да показва флагове или състояния.

struct packed\_struct {

unsigned int f1:1;

unsigned int f2:1;

unsigned int f3:1;

unsigned int f4:1;

unsigned int type:4;

unsigned int my\_int:9;

} pack;

Тук имаме 6 полета – 4 броя еднобитови, едно поле с 4 бита и едно с 9. Тук особеното е, че ако размера на структурата е с размерност до една дума с размер Int ще побере информацията в един Int, иначе ще използва две думи или повече.

# Дефиниране на собствен тип

Използваме ключовата дума ***typedef*** за да дефинираме наш тип данни. Това може да стане по два начина:

struct Distance{

int feet;

float inch;

};

Typedef struct Distance Distance

int main() {

Distance d1, d2;

}

По този начин не е нужно всеки път да пишем думата struct

или

typedef struct Distance {

int feet;

float inch;

} distances;

int main() {

distances d1, d2;

}

Други примери:

#include <stdio.h>

#include <string.h>

// struct with typedef person

typedef struct Person {

char name[50];

int citNo;

float salary;

} person;

int main() {

// create Person variable

person p1;

// assign value to name of p1

strcpy(p1.name, "George Orwell");

// assign values to other p1 variables

p1.citNo = 1984;

p1. salary = 2500;

// print struct variables

printf("Name: %s\n", p1.name);

printf("Citizenship No.: %d\n", p1.citNo);

printf("Salary: %.2f", p1.salary);

return 0;

}

# Влагане на структури

Като поле за структура може да имаме друга структура, стига да се вече дефинирана.

struct complex {

int imag;

float real;

};

struct number {

struct complex comp;

int integers;

} num1, num2;

Достъпа става по аналогичен начин:

num2.comp.imag = 11;

#include <stdio.h>

struct complex {

int imag;

float real;

};

struct number {

struct complex comp;

int integer;

} num1;

int main() {

// initialize complex variables

num1.comp.imag = 11;

num1.comp.real = 5.25;

// initialize number variable

num1.integer = 6;

// print struct variables

printf("Imaginary Part: %d\n", num1.comp.imag);

printf("Real Part: %.2f\n", num1.comp.real);

printf("Integer: %d", num1.integer);

return 0;

}

# Масив от структура

Масив от структура използваме по аналогичен начин на масив от променливи.

struct Distance{

int feet;

float inch;

};

Typedef struct Distance Distance

int main() {

Distance d1[10], d2;

….

d1[3].feet=23;

}

Не можем да използваме оператори +,- и подобни за елемент от тип структурата ни, понеже не е дефинирано използването на конкретния оператор с данните от структурата. За да ползваме полетата от структурата и да извършваме действия трябва да влизаме в тях и да стигаме до стандартните типове данни в езика.

Задачи:

1. Съставете структура за точка в пространството. Направете масив от точки и ги попълнете с данни. Направете триъгълници със зададените точки и намерете страните на триъгълниците.
2. Направете структура за превозно средство с подходящи полета. Направете втора структура за различен тип превозни средства и попълнете данните за променливи от вид кола, мотор лодка и самолет. Сравнявайте различните средства по характеристика по избор.
3. Направете масив от елементи с тип структура за ученици. Направете структура за випуск с няколко класа. Въведете данни за учениците от един клас и изчислете средния бал на групата. Изчислете средния бал на випуска.