Работа со структурами.

Рассмотрим структуру Person, содержащую 3 поля – имя, фамилию и возраст человека. Для возраста выберем тип int, а для хранения строк определим указатель на массив символов.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct Person

{

char\* name;

char\* surname;

int age;

};

int main()

{

struct Person Student1;

Student1.name = (char\*)malloc(25);

Student1.surname = (char\*)malloc(25);

strcpy(Student1.name, "Ivan");

strcpy(Student1.name, "Ivanov");

Student1.age = 25;

free(Student1.name);

free(Student1.surname);

return 0;

}

Для работы с данной структурой необходимо **выделить память** под хранение имени и фамилии. Для этого используется функция malloc(). Передача строк происходит с помощью функции strcpy().

Создадим указатель на данную структуру.

struct Person \*Student2;

Student2 = &Student1;

Student2->age = 26;

Обратите внимание, что работа через указатель ведется с помощью символа ‘**->**’. После выполнения данных команд значение возраста **Student1** изменится на 26, т.к. работа ведется через указатель.

|  |  |
| --- | --- |
| printf("Student1:age %d\n", Student1.age);  Student2->age = 26;  printf("Student1:age %d\n", Student1.age); |  |

Оформим части кода в виде функций.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct Person

{

char\* name;

char\* surname;

int age;

};

void printstudent(const struct Person S)

{

printf("%s\n", S.name);

printf("%s\n", S.surname);

printf("%d\n", S.age);

}

void StudentInit(struct Person \*S)

{

S->name = (char\*)malloc(25);

S->surname = (char\*)malloc(25);

S->age = 0;

}

void StudentAddInfo(struct Person\* St, const char\* N, const char\* S, int A)

{

strcpy(St->name, N);

strcpy(St->surname, S);

St->age = A;

}

void StudentDel(struct Person \*S)

{

free(S->name);

free(S->surname);

}

int main()

{

struct Person Student1;

StudentInit(&Student1);

StudentAddInfo(&Student1, "Ivan", "Ivanov", 25);

printstudent(Student1);

StudentDel(&Student1);

return 0;

}

Заметим, что на данный момент все функции существуют отдельно от структуры. Для удобства, возможно объединить их вместе с данными внутри структуры. Такая реализация возможна, но только в языке С++.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct Person

{

char\* name;

char\* surname;

int age;

Person()

{

printf("Default born!\n");

name = (char\*)malloc(25);

surname = (char\*)malloc(25);

age = 0;

}

Person(const char\* N, const char\* S, int a)

{

printf("Born!\n");

name = (char\*)malloc(25);

surname = (char\*)malloc(25);

strcpy(name, N);

strcpy(surname, S);

age = a;

}

~Person()

{

printf("Death %s :(\n", name);

free(name);

free(surname);

}

void addage()

{

age++;

}

void printstudent()

{

printf("%s\n", name);

printf("%s\n", surname);

printf("%d\n", age);

}

};

int main()

{

{

struct Person Student1("Ivan", "Ivanov", 25);

struct Person Student2;

Student1.printstudent();

strcpy(Student2.name, "Natalia");

strcpy(Student2.surname, "Petrova");

Student2.printstudent();

Student2.addage();

Student2.printstudent();

}

return 0;

}

Рассмотрим основные особенности данной реализации:

1. Функция инициализации по умолчанию

Person()

{

printf("Default born!\n");

name = (char\*)malloc(25);

surname = (char\*)malloc(25);

age = 0;

}

Название данной функции должно совпадать с названием структуры. Она будет вызываться в момент создания экземпляра нашей структуры.

struct Person Student2;

В ней происходит выделение памяти и инициализация возраста 0.

1. Функция инициализации с параметром

Person(const char\* N, const char\* S, int a)

{

printf("Born!\n");

name = (char\*)malloc(25);

surname = (char\*)malloc(25);

strcpy(name, N);

strcpy(surname, S);

age = a;

}

Аналогично функции инициализации по умолчанию, её название должно совпадать с названием структуры. Только в данном случае в функцию передаются параметры для заполнения структуры. Данная функция срабатывает при вызове функции с параметрами:

struct Person Student1("Ivan", "Ivanov", 25);

1. Функция уничтожения объекта структуры

~Person()

{

printf("Death %s :(\n", name);

free(name);

free(surname);

}

Название данной функции также должно совпадать с названием структуры, но начинаться с символа ~.

Её основная особенность заключается в том, что данную функцию не нужно вызывать напрямую из кода. Она вызывается автоматически при выходе объекта структуры из области видимости, т.е. при достижении закрывающей фигурной скобки. В данном примере область видимости завершается по окончании функции main.

Если дополнительно добавить скобки и уменьшить область видимости, то функция уничтожения выполнится раньше, чем выведется сообщение *Hello!*

|  |  |
| --- | --- |
| int main()  {  {  struct Person Student1("Ivan", "Ivanov", 25);  }  printf("Hello!\n");  return 0;  } |  |