Faculdades Integradas de Bauru - FIB

CONSTRUÇÃO DE ALGORITMOS E PROGRAMAÇÃO II
CAP II - Aula 8

Prof. Me. Marco Aurelio M. Antunes

Alocação Estática

Na alocação estática de memória, os tipos de dados tem tamanho predefinido. Neste caso, o compilador vai alocar de forma automática o espaço de memória necessário. Sendo assim, dizemos que a alocação estática é feita em tempo de compilação. Este tipo de alocação tende a desperdiçar recursos, já que nem sempre é possível determinar previamente qual é o espaço necessário para armazenar as informações. Quando não se conhece o espaço total necessário, a tendência é o programador exagerar pois é melhor superdimensionar do que faltar espaço.

Alocação Dinâmica

Na alocação dinâmica podemos alocar espaços durante a execução de um programa, ou seja, a alocação dinâmica é feita em tempo de execução. Isto é bem interessante do ponto de vista do programador, pois permite que o espaço em memória seja alocado apenas quando necessário. Além disso, a alocação dinâmica permite aumentar ou até diminuir a quantidade de memória alocada.

```c
#include <stdio.h>
#include <stdlib.h>
void main() {
    // Vetor estático para armazenar notas de 5 alunos
    int notas[5],i;

    // Preencha o vetor com notas
    for (i=0; i<5; i++) {
        printf("Digite a nota do aluno ");
        scanf("%i", &notas[i]);
    }

    // Exiba as notas
    printf("Notas dos alunos: ");
    for (i=0; i<5; i++) {
        printf("%i ", notas[i]);
    }

    getchar;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
void main() {
    int *notas=NULL;  // Inicialmente, o vetor dinâmico está vazio
    int numalunos,i;

    // Solicita o número de alunos
    printf("\nQuantos alunos deseja inserir notas? ");
    scanf("%i", &numalunos);

    // Alocação dinâmica de memória com base no número de alunos
    notas = (int *)malloc(numalunos * sizeof(int));

    if (notas == NULL) {
        printf("\nFalha na alocação de memória");
        return(0);
    }

    // Preencha o vetor dinâmico com notas
    for (i=0; i<numalunos; i++) {
        printf("Digite a nota do aluno");
        scanf("%i", &notas[i]);
    }

    // Exiba as notas
    printf("Notas dos alunos: ");
    for (i=0; i<numalunos; i++) {
        printf("%i ", notas[i]);
    }

    // Liberação da memória alocada
    free(notas);
}
```

A função malloc

A função malloc (o nome é uma abreviatura de memory allocation) aloca espaço para um bloco de bytes consecutivos na memória RAM (= random access memory) do computador e devolve o endereço desse bloco. O número de bytes é especificado no argumento da função. No seguinte fragmento de código, malloc aloca 1 byte.

A função sizeof

A função sizeof é usada para se saber o tamanho de variáveis ou de tipos. Ele retorna o tamanho do tipo ou variável em bytes. Mas porque usá-lo se sabemos, por exemplo, que um inteiro ocupa 2 bytes? Devemos usá-lo para garantir portabilidade. O tamanho de um inteiro pode depender do sistema para o qual se está compilando.

A função free

As variáveis alocadas estaticamente dentro de uma função, também conhecidas como variáveis automáticas ou locais, desaparecem assim que a execução da função termina. Já as variáveis alocadas dinamicamente continuam a existir mesmo depois que a execução da função termina. Se for necessário liberar a memória ocupada por essas variáveis, é preciso recorrer à função free.

A função free desaloca a porção de memória alocada por malloc. A instrução free (ptr) avisa ao sistema que o bloco de bytes apontado por ptr está disponível para reciclagem. A próxima invocação de malloc poderá tomar posse desses bytes.

structs (Estruturas)

As structs são usadas para criar tipos de dados personalizados que podem conter um ou mais campos (variáveis) de diferentes tipos. As structs são úteis quando você deseja representar um conjunto de dados relacionados sob um único nome.

Representação de Registros de Dados: As structs são frequentemente usadas para representar registros de dados. Por exemplo, você pode usar uma struct para representar informações sobre um funcionário, um aluno, um produto, etc. Cada campo da struct pode conter informações específicas, como nome, ID, salário, etc.

```
struct funcionario {
    int id;
    char nome[50];
    float salario;
};
```

```c
struct produto {
    char nome[50];
    float preco;
    int quantidade;
};

struct produto produtos[5];  // Cria um vetor de 5 produtos

// Leitura das informações dos produtos
for (i=0; i<5; i++) {
    printf("Digite o nome do produto: ");
    scanf("%s", produtos[i].nome);
    printf("Digite o preço do produto: ");
    scanf("%f", &produtos[i].preco);
    printf("Digite a quantidade do produto em estoque: ");
    scanf("%d", &produtos[i].quantidade);
}

// Exibição das informações dos produtos
printf("\nInformações dos Produtos:\n");
for (i=0; i<5; i++) {
    printf("\nProduto %i:",i);
    printf("\nNome: %s", produtos[i].nome);
    printf("\nPreco: R$%.2f", produtos[i].preco);
    printf("\nQuantidade em Estoque: %i", produtos[i].quantidade);
    printf("\n");
}
```

Dúvidas ????

Exercícios

1 – Fazer um programa que leia a idade, o peso e a altura de 10 funcionários. Fazer os cálculos necessários, mostre o dados lidos, o total e a média de altura, peso e idade dos funcionários.

2 – Fazer um programa que leia o quantidade de km a ser percorrida e a quantidade de km que um carro faz com 1 litro de combustível. Fazer os cálculos qkmpercorrida[x] / consumo[x] e informe quantos litros de combustível são necessários para percorrer a distância. Fazer isso com um vetor para 6 veículos;

3 – Utilizando Struct, escreva um programa que leia o id, o nome e o salário de 8 funcionários e, em seguida, pergunte ao usuário por um id. O programa deve verificar se o id existe no vetor e informar se foi encontrado ou não. Se foi encontrado mostrar o nome e o salário do funcionário, senão mostrar mensagem informando o erro.

4 – Fazer um programa que leia o preço de 10 produtos. Leia a cotação do dólar e a cotação do euro uma ÚNICA vez. Mostre os preços dos produtos lidos e seus respectivos valores em dólar e euro.

5 – Utilizando vetor dinâmico, leia o nome e a média fina de N alunos. Mostre o nome e a média final de todos os alunos e verifique se a média final for maior que 7 o aluno está aprovado, senão o alunos está de exame.

6 – Fazer um programa que leia a temperatura de uma cidade dos últimos 15 dias. Utilizar Climatempo para fonte de dados verídicas. Mostrar todas as temperaturas, a maior temperatura, a menor temperatura e a temperatura média.